DOCUMENT RESUME

ED 168 574 IR 007 115

TITLE	Sixth New Zealand Computer Conference (Auckland 78). Volume I, Papers.
INSTITUTION	New Zealand Computer Society, Auckland.
PUB DATE	78
NOTE	537p.; Not available in hard copy due to small type size of the original document; For related document, see IR 007 116
AVAILABLE FROM	New Zealand Computer Society, National Executive Office, P.O. Box 2123, Wellington (Vol. I $6.00 and Vol. II $4.00)
EDRS PRICE	MF02 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	Computer Assisted Instruction; *Computer Oriented Programs; Computers; *Computer Science; Dial Access Information Systems; Discussion Programs; Electronic Data Processing; Information Networks; *Information Processing; Programing Languages; *Technological Advancement; Telecommunication

ABSTRACT

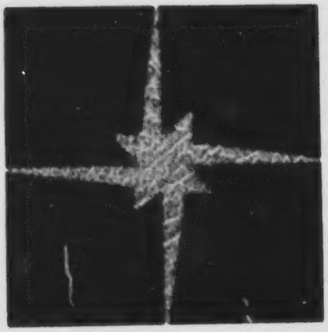
This collection of conference presentations includes 23 papers on a variety of topics pertaining to the use of computer in New Zealand. Among the topics discussed are computer science techniques in a commercial data processing situation, data processing personnel and their careers, the communication aspects of an airline system, implementation of Alsol-W programming systems, computer communications and the fire service, observations on user/vendor communications, future database and application development techniques, system control via a high level language, the organization and management of systems development, and human communication between educational data processors and those outside the profession, e.g., the users. (JEG)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED168574

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY



SIXTH
NEW ZEALAND
COMPUTER
CONFERENCE
AUCKLAND 78

IR007115

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

C.C. Bradley

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

PAPERS VOLUME I

CONTENTS BY AUTHOR

AUTHOR	TITLE OF PAPER	PAGE NO.
BIELESKI, I. P.	*Computer Science Techniques In A Commercial Data Processing Situation*	1
BOSWELL, C. R. (Dr) & MELHUISH, P. J.	*Data Processing Personnel & Their Careers – A Profile*	20
	&	
	Teaching Of Computing In NZ – The Dream And The Reality	54
BOWLES, S. W. (Dr) & COLLEAGUES	*VicNet – An X25 Packet Switching Network*	83
CAMPBELL, IAN & PAVER, LES	*Communication Aspects Of An Airline Reservation System*	144
COLLINS & HUNT, L. R.	*The Implementation Of Alsol-W Programming Systems For N.Z. Schools*	185
COOKE, M. P.	*A Friendly Terminal – Why Not?*	204
COOPER, R. E. M. (Dr)	*Computer Communications And The Fire Service*	238
DONALD, B. R.	*Observations On User/Vendor Communications*	253
FINKLESTEIN, C. B.	*Future Database And Application Development Techniques*	262
JACKSON, I. F. (Dr)	*A Relational Model Of An Accounting System*	292
LAWSON, R. P.	*System Control Via A High Level Language*	324
LENS VAN RIJN, J. A.	*Communication: From Systems Requirements To Systems Architecture*	347
MARKS, T. W. (Dr)	*Centralised Communications For Melbourne Fire Brigade*	358
McDOUGALL, N. N.	*Management Consultant – Aid To Effective Communication Or Expensive Luxury*	396
MITCHELL, K. I.	*Specification To Implementation In 24 Hours!*	411
NORRIS, P.	*Some Thoughts On Computers And Languages*	429
PAYNE, A.	*The O & M Of Systems Development*	440
SAY, B. A.	*Health Computer Project*	457
SPENCE, R. J.	*Fleshware And The Communication Problem*	466
STUBBS, J. (Presented by NOEL MURPHY)	*Business Forms And Business Forms Management Programs*	485
ZEN, LOY (Dr) & RIDWAN, R. R.	*Describing The Real World For The Data Base*	504
HANSON, O. J.	*Choice Of File Organisation*	517

CONTENTS BY PAPER TITLE

TITLE	AUTHOR	PAGE NO.
A Friendly Terminal – Why Not?	COOKE, M. P.	204
A Relational Model Of An Accounting System	JACKSON, I. F. (Dr)	292
Centralised Communications For Melbourne Fire Brigade	MARKS, T. W. (Dr)	358
Choice Of File Organisation	HANSON, O. J.	517
Communication Aspects Of An Airline Reservation System	CAMPBELL, IAN & PAVER, LES	144
Communication: From Systems Requirements To Systems Architecture	LENS VAN RIJN, J. A.	347
Computer Communications And The Fire Service	COOPER, R. E. M. (Dr)	238
Computer Science Techniques In A Commercial Data Processing Situation	BIELESKI, I. P.	1
Data Processing Personnel & Their Careers – A Profile	BOSWELL, C. R. (Dr) & MELHUISH, P. J.	20
Describing The Real World For The Data Base	ZEN, LOY (Dr) & RIDWAN, R. R.	504
Fleshware And The Communication Problem	SPENCE, R. J.	466
Future Database And Application Development Techniques	FINKLESTEIN, C. B.	262
Health Computer Project	SAY, B. A.	457
Improving Communication Between The Analyst And The End User In The Systems Design Process	JENKIN, B. G.	309
Management Consultant – Aid To Effective Communication Or Expensive Luxury	McDOUGALL, N. N.	396
Observations On User/Vendor Communications	DONALD, B. R.	253
Some Thoughts On Computers And Languages	NORRIS, P.	429
Specification To Implementation In 24 Hours!	MITCHELL, K. I.	411
System Control Via A High Level Language	LAWSON, R. P.	324
Teaching Of Computing In N.Z. – The Dream and The Reality	MELHUISH, P. J. & BOSWELL, C. R. (Dr)	54
The Implementation Of Alsol-W Programming Systems For N.Z. Schools	COLLINS & HUNT, L. R.	185
The O & M Of Systems Development	PAYNE, A.	431
Vicnet, – An X25 Packet Switching Network & Others	BOWLES, S. W. (Dr) & COLLEAGUES	83

COMPUTER SCIENCE TECHNIQUES IN A COMMERCIAL DATA PROCESSING SITUATION

Author: I. P. Bieleski, M.B.C.S., M.N.Z.C.S.
Senior Lecturer
Department of Computer Science
Massey University
PALMERSTON NORTH

This paper describes a commercial computer application that appeared to be of classic simplicity. In fact the real parameters of the system posed a few problems. The solutions to these problems were not found by application of accumulated experience but rather via the application of recently acquired knowledge of computer science topics such as taught in New Zealand Universities. The requirements of a very extensive edit program delivered to a tight deadline was solved using top down and structured programming techniques. The problem of manufacturing many individual programs to very similar specifications was solved by the reshaping of the base language into another form by the use of macro-isation.

The problem of carrying a large number of totals during a tabulation was solved by the technique of treating the data as a tree structure and performing and end-order traversal of the totals stored on disk. Comparative performance of this method is established.

Key words and phrases: PL/I, stack, RPG, tabulation, top down, structured programming, macros.

CR Categories: 4.12, 5.32

Introduction

This paper describes the techniques involved in the design of a computer system to capture, edit, and summarise some statistical data. It resulted from a contract being signed when a considerable amount of development work had already been undertaken by the client. There was limited resources available for the work while the due out date was fixed.

With the major responsibility for the work, the author selected the priorities for the project and worked accordingly. They were in order:

1. Programs working to specification.
2. On time installation.
3. All programs to fit into a 140k partition of an IBM 360/50.
4. Programs must be easy to adapt for future changes.
5. Efficient cost effective programs.

There were obviously two phases: The capture of valid data to be run periodically while the punching was undertaken by the client and the running of the summarising and reporting process at the end.

This suggested the classic suite of programs.

1. A program to capture batches of input data.
2. A sort program to sort the data into the report sequence.
3. An edit and update program to maintain a sequential file and produce an error report.
4. Report programs to summarise and tabulate the results.

Rather than show this in the familiar flowchart style, I have pictured this in Fig. 1 with a state transition diagram that shows different states of the data during the project.

Legend

1. Tape encoding
2. Data capture program
3. Edit/update program
4. Clerical
5. Report programs

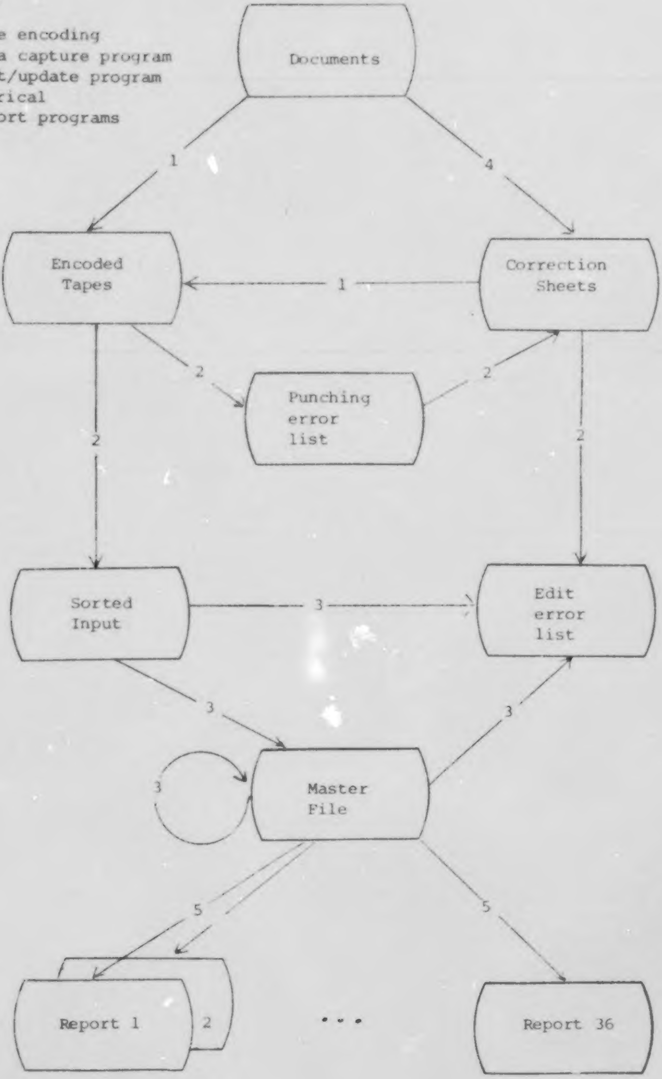

Documents
1
4
Encoded Tapes
1
Correction Sheets
2
2
Punching error list
2
2
Sorted Input
3
Edit error list
3
3
3
Master File
5
5
Report 1
2
...
Report 36

Figure 1 Data State Transition Diagram

Reality

The real parameters of the problem posed some particular implementation problems that are worthy of discussion. The raw data was already prepared in the form of an expected 70,000 documents, each of 20 pages of statistical information. The data consisted of two identifying fields of data plus some 332 items with up to 9 fields of numeric data each. This amounted to 1,362 fields some of which could be as much as 7 digits long.

The data had to be classified by program to develop several new fields of identifying data. The data had to be edited extensively. Over 100 classes of error were specified and resulted in some 660 programmed checks being made.

For the reporting phase, some 36 reports were specified that would result in 60 output listings. These were to contain totals, frequencies, averages and metric conversions.

The most particular problem arose from several reports which required a tabulation with four levels of total (county, province, island and grand total) with, instead of the usual one line total, a type by size matrix of 36 x 22 of results. With up to sixteen columns per print out this would require a total of 50,688 counters to accumulate these during the tabulation.

The standard manufacturers courses in programming etc. did not give sufficient grounding to handle all the problems represented here.

Data Capture

With the potential number of input fields amounting to some 95 million fields, the punching of redundant fields had to be avoided. The client had done the design and planning in this area already. A new key-to-tape system was being installed. The input data was to be punched on an item by item basis. In practice 35 out of 332 items were punched on average, although no prediction was available. With the possibility of up to 7 digit fields the need to punch leading zeroes was avoided by going to free form input. The two identifying fields together with the identifying item number were punched in fixed leading fields while the significant digits of the up to 9 fields were punched in free form with a single blank separator. A special character completed the line. No zero was punched for a zero field, just another delimiting blank.

The handling of free form data is something that university computer science courses prepare students for, but not so in manufacturers' programming courses. However this problem being at the simplest possible level, did not cause any difficulty whatever. The input program that did the conversion of free form data to binary was combined with the sorting process as an exit from the standard sort package.

The only problems in this area were due to the lack of a planned system of batch controls and the need to allow for the by-pass of badly recorded tape blocks.

Language

The author's choice was PL/I due to proficiency and scope. Although the service bureaux's standard language was COBOL, the need for the full range of facilities was accepted. This was born out in practice. The input capture program was costed on the basis of programming in COBOL yet the PL/I program turned out to be a mere 110 statements including the sort exits and was considerably cheaper than estimated to get going. This helped a little in overcoming internal resistance in the bureaux to the planned change to PL/I. The ability of the Optimising compiler to produce efficient programs saved the bother of having to consider efficiency while coding.

File Design

The master file record contained, in addition to the two identifying fields and the 1362 item fields, key fields of province and island plus classifications of type and various size categories determined in the edit process. A couple of spare empty fields were added just in case a use might arise for extra fields. The total length of all this was 5484 bytes. Rather than use multiple trailer records for item or groups of item records, the data was recorded as compressed records of variable length with each non-zero field being stored with a single field that identified the item and subscript numbers of the field. This was written as variable length spanned records.

This reduced the file size from a potential 10 reels at 1600 bpi to just one reel at the expense of some additional time expanding and compressing the records using a field location table. There was no data available to predict just how much would be saved in file space, so the reduction to a 7% figure was gratifying.

Edit Program

The size of the specifications for the editing and classification of the data seemed a bit overwhelming but recent exposure to the ideas propounded by a travelling advocate of structured programming from Massey University inspired the author to try this new approach. The program was developed in top down fashion with a start on the updating logic together with master file expansion contraction code. This was tested in conjunction with the input capture program. The coding of the classification and editing followed step by step and added to a running program and was able to be tested in a sound environment.

The time schedule for this particular program was critical as the capture and editing cycle was done over a period of months due to the volumes involved.

Although the standard of specification was very high and the resultant program size at 900 statements and 1400 cards not as high as estimated, the complexity of some of the classification logic needed great care. One classification test involved the use of a logical expression that contained 91 logical and arithmetic operators. This one statement produced over 1500 bytes of code. Significant use was made of the advanced features of PL/I to write compact and understandable (in terms of the specification) source code. Reliance was placed on the PL/I Optimising compiler to make the object code efficient rather than waste time on tuning and hand optimisation. The statements averaged 77 bytes of object code each.

The specifications such as "Definite error 31. An entry in item 40 and no entry in either item 001 or 002." were such that they could be coded straight off into PL/I code:

```
    E31: IF ((F040(1) > 0) | (F040(2) > 0)) & (F001(1) = 0) &
              (F002(1) = 0) THEN CALL ERROR (31,40,F040);
```

With a total of 660 error checks and 10 classifications just ticking off each one in turn was a significant amount of work. This was greatly helped by the ability of the client's specifying analyst to read the code and verify it, even though he had no formal training in PL/I. This resulted in several bugs being discovered before individual testing and several ambiguous definitions being respecified.

Consider this specification: "Code 8. When the combined entries in items 18 plus 19 are between and including 51 and 74 (but the entry in item 19 does not exceed 25) and there is an entry in any of the items 20, 22 to 39 of between and including 20 and 40 <u>or</u> where the entry in item 18 is between and including 51 and 74 and there is an entry in item 19 of between and including 25 and 40".

The testing proceeded systematically and smoothly. The built-in debugging features also came into play occasionally. Altogether there was no problem.

After the first production cycle it was necessary to alter some edit checks in the light of the real characteristics of the data.

In all, each record had to pass completely through 55k of code including many loops. Hence, the performance of 2½ records edited per second was considered very good. Initially the program just met the storage requirements but the amendments and corrections being applied during the running of the system resulted in the criteria of 140k not being met in the final versions.

Reports

Once the coding of the edit checks within the edit and update program was well under way the problems of producing the report programs could be tackled. As the 36 reports had been designed with a 160 print position printer in mind, many of the report layouts which were often already in several parts, had to be redesigned. The total number of printfiles required thus became 60. With an additional worker becoming available the author was left with the responsibility for reports 1 to 9. This required 26 print files. These nearly all contained the need to keep the 36 by 22 type by size matrix for each column to be printed and repeated for each of the four levels of total of the tabulation.

Sixteen columns would require 50,688 counters or 202,752 bytes which was a bit beyond the 140k design point. While the production run could be specially run in a large partition the problem of creating and testing 26 report runs within 140k remained.

My colleague was an experienced COBOL man and had already started on his set of reports which included the 36 by 22 matrix problem. He solved his problems independently and his methods are not discussed in this paper.

These two problems of working space and multiple reports were tackled together, however the methods used can be treated separately.

Report Program Manufacture

As all the tabulations discussed here were based on the same control breaks, attention was first given to the overall structure of the programs. My newly acquired knowledge of structured programming was applied to the fundamental design of a tabulation process. A properly structured framework was developed without any GOTO's and which had a good top down flow.

The components of this report independent structure was made into PL/I macros and catalogued in a source library. The report specific code could then be inserted in between each component of the skeleton which could be copied in from the library giving a semblance of a RPG structure. This is illustrated in Fig. 2.

The macro SETUP contained the declarations for master file record in compressed and expanded form, master file, and expansion index table. It opened the file and obtained the first record.

Macro DETAILS expanded each compressed record into the expanded form and saved the key fields of county, province, and island for testing and control breaks.

Macro PROGL1 read the records and passed control onwards for the first level of control break.

Macro PROGL2, PROGL3, and PROGL4 passed control onwards for levels 2 to 4.

Macro EOJRTN printed end of job accounting and controls, closed the master file and closed inherited loops.

Each report program was developed by inserting the particular code within this framework and testing it independently. The output layout had to be got exactly right, and the computations such as averages, metric conversions etc. had to be debugged. The overall logic did not have to be debugged again because it was fixed in the macro structure.

Once the framework was established it was possible to code the details of the program (apart from the specification of the page headings) at the keypunch and prepare the program in an hour or two. These programs were tested in 140k partitions.

For the final production runs, the segments of the programs were mixed and matched in suitable combinations and inserted within the standard framework. Thus six large programs were produced and the 26 reports produced in six passes of the master file.

13

```
REPORT1: PROCEDURE OPTIONS(MAIN) REORDER;
     %INCLUDE SETUP;
     /* Declare counters, page overflow routine,
        setup first page.                         */
     .
     .
     .
     %INCLUDE DETAILS;
     /* Details calculations to accumulate
        specified results.                        */
     .
     .
     .
     %INCLUDE PROGL1;
     /* Level 1 calculations.
        Level 1 print.                            */
     .
     .
     .
     %INCLUDE PROGL2;
     /* Level 2 calculations.
        Level 2 print.                            */
     .
     .
     .
     %INCLUDE PROGL3;
     /* Level 3 calculations.
        Level 3 print.                            */
     .
     .
     .
     %INCLUDE PROGL4;
     /* Level 4 calculations.
        Level 4 print.                            */
     .
     .
     .
     %INCLUDE EOJRTN;
     END REPORT1;
```

Figure 2 Report program framework

Source Code Manipulation

An additional minor use was made of the PL/I preprocessor facilities. Report 9 required a total of 10 print files to be produced, each with very similar calculations required. One program was written together with some macro statements to vary the program text. This program was compiled and the ten variations required were punched out in source text form by the preprocessor facilities.

Of course with a modern interactive system available this could be just as easily done with text editor facilities.

Saving Storage

The problem of economising on the use of storage for the maintaining of four matrices of totals was obviously going to be solved by storing data on disk. Exactly how this was going to be done was not clear when the problem was discovered while studying the documentation.

At about this time the author as part of a self education process, had discovered Knuth's book of Fundamental algorithms.

The standard topics of computer science courses such as lists and trees were quite revealing to some one educated with just manufacturers courses. Some experimentation soon brought about some proficiency with tree scanning techniques such as "pre-order traverses".

This background provided the insight to see that if one treated the data structure as a tree instead of a sequential list then a tree scanning process using a stack of totals could be used to solve the problem.

The data could be represented as the tree in Fig. 3. The total at any node is the sum of the nodes below it in the tree. So an end-order traverse of the tree would produce the required totals in the required sequence. The way to traverse the tree would be to use a stack (or LIFO list).

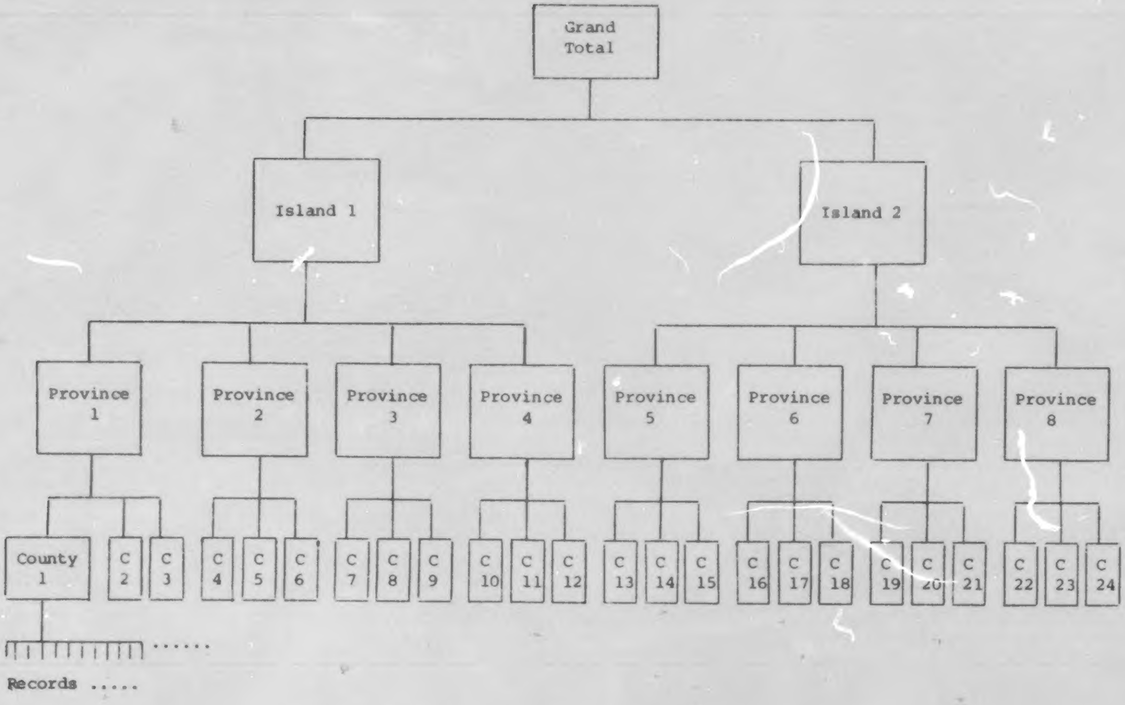

Grand Total
Island 1
Island 2
Province 1
Province 2
Province 3
Province 4
Province 5
Province 6
Province 7
Province 8
County 1
C 2
C 3
C 4
C 5
C 6
C 7
C 8
C 9
C 10
C 11
C 12
C 13
C 14
C 15
C 16
C 17
C 18
C 19
C 20
C 21
C 22
C 23
C 24
Records .....

Figure 3 Tree Structure of the Tabulation

The data would still be accessed in the conventional sequential way but with the role of the scanning stack reversed to be the collection of totals. Only one matrix for the totals would be required instead of four.

The Method

The terminal nodes (records) were scanned by reading the file in sequence and taking the control breaks in the conventional manner. On each level 1 break (change of county), besides printing the matrix, the matrix was written out in up to 36 segments forwards on the disk. This corresponds to the "push" operation on the stack. Each segment contained 22 rows of totals and was written out, provided the totals were not all zero. Each segment contained identifying island, province, county and type fields. These were used to control the retrieval of segments.

A dummy segment was initially written with nines in the key fields so that an end of stack condition would be indicated.

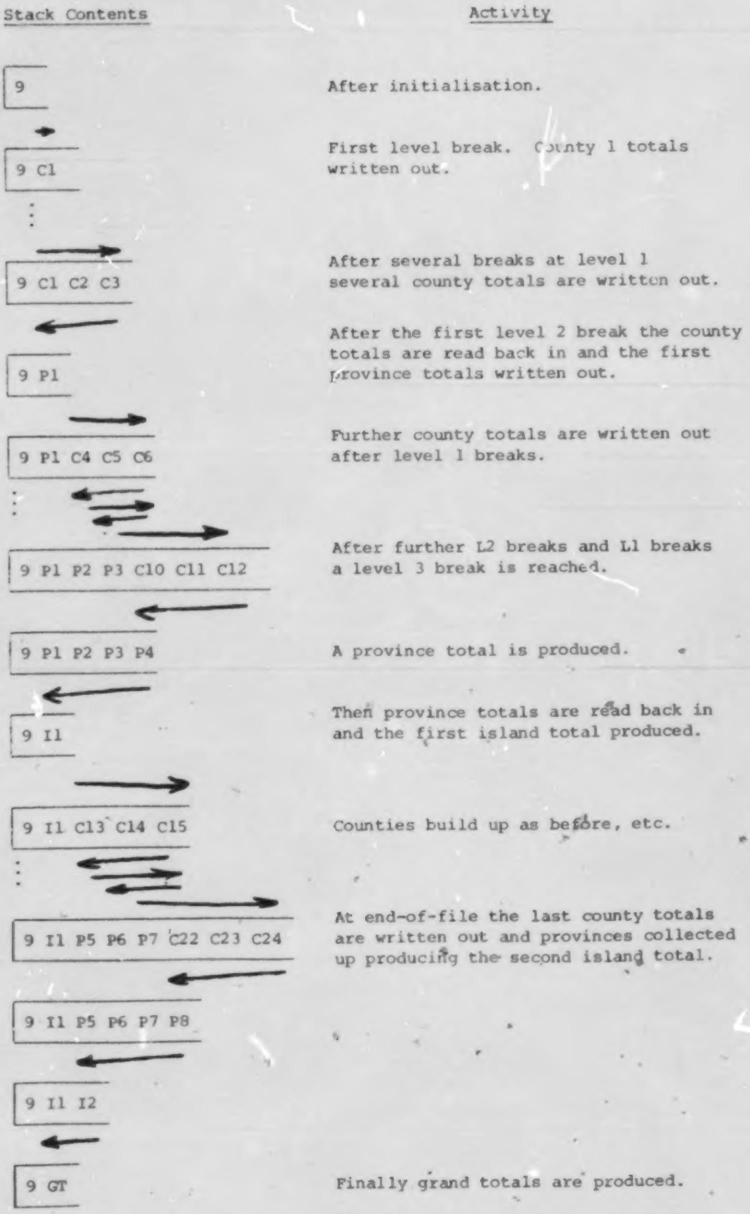
When a level 2 break (change of province) occurred, then, following the printing and writing to the stack for the implied level 1 break, the disk file was read backwards ("pop" operation) and level 2 totals accumulated for that province using the same matrix of counters. Level 2 totals could then be printed while at the same time writing out the level 2 totals going forwards in a "push" operation again.

The stack contents are illustrated in Fig. 4.

The same procedure is extended for the level 3 break (island change) with "pop" operations allowing the collection of the level 2 totals and a "push" operation to write the level 3 totals.

Similarly for the level 4 grand totals signalled by the end of file. The "push" operation for the grand total is redundant.

The process is thus an "end-order" traverse of the tree structure.



Stack Contents	Activity
9	After initialisation.
9 C1	First level break. County 1 totals written out.
9 C1 C2 C3	After several breaks at level 1 several county totals are written out.
9 P1	After the first level 2 break the county totals are read back in and the first province totals written out.
9 P1 C4 C5 C6	Further county totals are written out after level 1 breaks.
9 P1 P2 P3 C10 C11 C12	After further L2 breaks and L1 breaks a level 3 break is reached.
9 P1 P2 P3 P4	A province total is produced.
9 I1	Then province totals are read back in and the first island total produced.
9 I1 C13 C14 C15	Counties build up as before, etc.
9 I1 P5 P6 P7 C22 C23 C24	At end-of-file the last county totals are written out and provinces collected up producing the second island total.
9 I1 P5 P6 P7 P8	
9 I1 I2	
9 GT	Finally grand totals are produced.

Figure 4 Stack Contents

Stack Implementation

The stack was obtained by the use of a PL/I REGIONAL (1) file which uses a relative key. During the "push" operation 1 was added to the key for every segment written while 1 was subtracted for every "pop" operation reading a segment.

Very little extra code was required to implement the method. It amounted to 3 declaration statements, 4 initialisation statements, 7 statements for the "push" operation in the print routine plus 7 statements for the three higher level control breaks for the "pop" operation.

The logic for the "push" and "pop" operations are shown as structure diagrams in Figs. 5 and 6.

Evaluation

The methods used were entirely successful. The RPG style program enabled each report program to be tested independently, while the stacking of totals on disk saved the space taken by 3 matrices less the space taken by the extra code and a segment I/O area. The overhead of writing and reading the segments was not excessive when considering the final file volume of 65,000 records.

There were approximately 300 matrix writes and reads.

When it came to the actual tabulations after the file was created and all errors corrected, the problems of scheduling a large number of print runs in the normal partition size had to be compared with doing more reports at a time using a difficult to schedule large partition. Putting two reports together meant that the total storage for the combined program was 50k less than the sum of the independent programs. As well, the passing of the file and the expansion of records was reduced. It was obvious that the cost of multiple reports in one pass was less.

By careful accounting of the storage requirements of the individual reports, combinations were worked out that put as many reports together

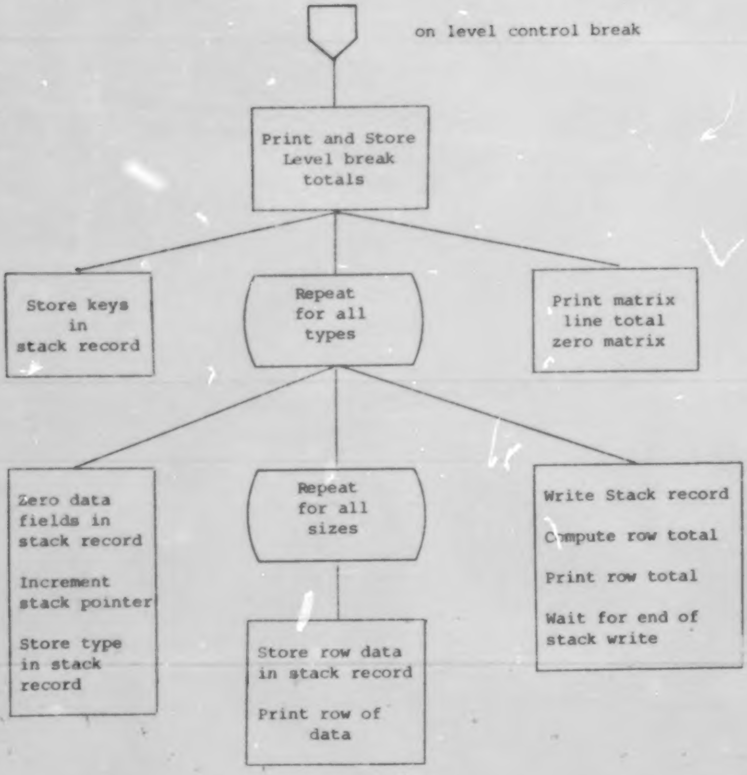

on level control break
Print and Store Level break totals
Store keys in stack record
Repeat for all types
Print matrix line total zero matrix
Zero data fields in stack record
Increment stack pointer
Store type in stack record
Repeat for all sizes
Write Stack record
Compute row total
Print row total
Wait for end of stack write
Store row data in stack record
Print row of data

Figure 5 "Push" Operation on Stack

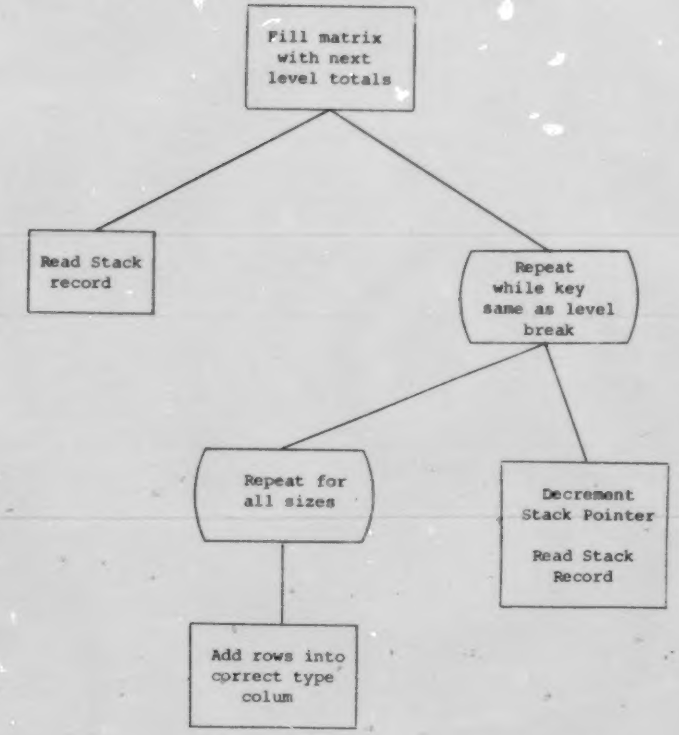

Fill matrix with next level totals
Read Stack record
Repeat while key same as level break
Repeat for all sizes
Decrement Stack Pointer
Read Stack Record
Add rows into correct type colum

Figure 6 "Pop" Operation on Stack

as possible. Six programs were thus produced which output either four or five reports at a time within a 300k partition.

The components of the selected report programs were inserted within the standard skeleton as in Fig. 2. The identifiers used within the report specific code has been arranged so that there was no conflict and could be combined in this way right from the beginning.

With the problems being solved independently, a comparison can be made. There were eleven comparable programs. They were written in COBOL and used indexed sequential accessing to store the totals. Each program produced one report only.

These programs were run in pairs using an extra copy of the file and multiprogramming in separate partitions.

Table 1 shows the statistics for the six PL/I and the eleven COBOL programs. When reducing these results to a pro-rata basis for the three coot components of storage, elapsed time, and CPU time the results are worthwhile in every respect. These are shown in Table 2.

Conclusions

The usefulness of some computer science knowledge when designing and programming commercial systems was demonstrated. The real needs of systems can give rise to complex considerations that warrant a broad education including some esoteric areas for which a practical use is not immediately apparent.

Reference

Knuth The Art of Computer Programming, Vol. 1, Fundamental Algorithms.

Table 1 Program Comparison

	PL/I Single Report Programs	PL/I Multiple Report Programs	COBOL Single Report Programs
Approximate number of source cards	250	950	1600
Storage range	60-158k	234-270k	136-142k
Average storage		256k	138k
Elapsed time range		107-121 mins	83-110 mins
Average elapsed time		116 mins	101 mins
CPU time range		61- 77 mins	38- 51 mins
Average CPU time		70 mins	45 mins

Table 2 Pro-rata Cost Factors

	PL/I Suite	COBOL Suite
Storage per report	59k	138k
Elapsed time per report	27 mins	50 mins
CPU time per report	16 mins	45 mins

DATA PROCESSING PERSONNEL AND THEIR CAREERS - A PROFILE

AUTHORS C.R. BOSWELL
Director
Computing Services Centre,
Victoria University of Wellington,
Wellington

and

P.J. MELHUISH
Senior Lecturer
Department of Computer Science
Massey University
Palmerston North.

ABSTRACT

The results of a New Zealand-wide survey of people employed in the areas of programming, systems analysis and data processing management are reported. This survey was supported by funds made available by the New Zealand Computer Society and Massey University. Responses were obtained from a complete cross-section of firms involved in data processing activities in New Zealand.

Items analysed were: age, sex, years of experience in the data processing field, educational qualifications, memberships of professional bodies, job satisfaction, type of employment and career expectations. In addition respondents gave details of their current annual salary, the annual value of other perquisites (car allowance, superannuation, etc.) and an estimate of any other supplementary income they earned from computing. Results for these, categorised by job title, are presented.

Respondents were asked to indicate their career path in data processing. A number of job descriptions were provided and people indicated the positions they had occupied, in order, and the time spent in each position. To avoid ambiguities inherent in job titles, specific descriptions of job attributes were provided for a number of categories and people were asked to use that job description that most nearly described their activities. These data have been analysed and typical career paths are presented.

Results from this survey are compared with those presented by the authors in a paper to the fifth New Zealand Conference of the New Zealand Computer Society in 1976.

KEYWORDS

Data processing careers, Survey of data processing staff, Salary scales, Job satisfaction, Career paths.

C.R. CATEGORIES

1.9, 2.2, 2.9

1. INTRODUCTION

This paper presents the results of a New Zealand wide survey of people involved in data processing. It included respondents employed in the management, systems analysis and programming fields.

The survey was sponsored by research grants from the New Zealand Computer Society and Massey University and is an extension of the work reported at the 5th New Zealand Conference of the Computer Society in Hamilton in 1976 (1). In the previous study selected firms co-operated in distributing a questionnaire to their staff. For this study management of 2,500 New Zealand firms known to use or have access to computing establishments were approached and asked to co-operate by distributing a questionnaire to their data processing staff. Approximately 2,000 questionnaires (Appendix 1) were distributed and 875 returned. After validation of the keypunched data 800 responses were used for subsequent statistical analysis. These have been analysed and the results are reported in this paper.

The questionnaires were distributed towards the end of 1977 and were returned in early 1978. Responses therefore indicate the conditions applying in December 1977 to January 1978. Data obtained from the questionnaire comprised:

Personal details: age, sex, time spent in data processing
Educational qualifications
Professional associations
Income
Job satisfaction
Employer description
Career expectations
Career developments.

For details, see Appendix 1.

A major problem encountered in administering the first questionnaire in 1975 was in the vexed area of job description. The decision made at that time was to give a series of job titles and leave the respondents to classify themselves. No guidelines were given explaining the job titles to help them in this. It has been suggested that there is a risk that the job title a respondent gives himself may not be that which would be given by his supervisor.

To try to reduce the possibility of this happening in the survey reported here brief descriptions of the qualifications and typical responsibilities corresponding to each job title used in the survey were included in the questionnaire.

This section was adapted from the Wellington IBM Users' Group Salary Survey form by the inclusion of a number of new job descriptions. By using and extending what is becoming a de facto New Zealand standard job specification it is hoped that the results reported here will have a wider applicability than a completely new set of descriptions. In addition, spare unlabeled boxes were provided for those who considered their job could not be categorised by any of the titles provided.

The Operations area was not considered in this particular survey. This is not because of any imputed unimportance of operations staff or a lack of interest on the part of the authors. Rather, it was decided that the difference in career structure and career paths for operations staff differs so markedly from others involved in data processing as to make a joint investigation unwieldy.

2. A PROFILE OF DATA PROCESSING PERSONNEL, 1977/78

2.1 Personal Details

2,1,1 Age

The age contribution of the respondents is given in Figure 1.

The data processing profession is clearly one for people in the age range 24-36. The proportion of people in this age range is 64% of those invloved in data processing, while 26% are 24-27. Half of the respondents were less than 30.

Figure 1: Age Distribution of People in Data Processing.

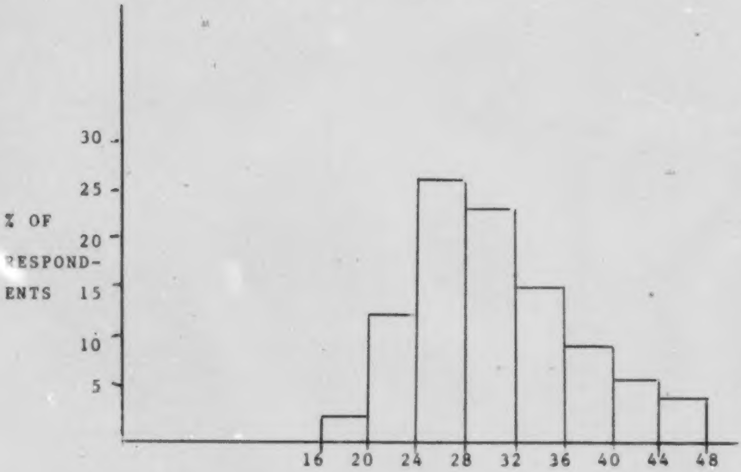


Table 1 gives the range of ages of respondents categorised by the title of the position they are currently holding.

2.1.2 Sex Distribution

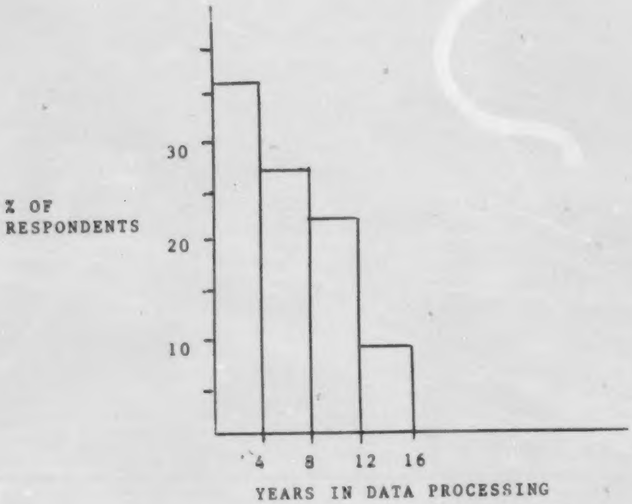
Data processing in New Zealand is a predominantly male field. Men comprised 82% of the respondents and women 18%. (There appears to be evidence that a degree of uncertainty is entering the data processing profession. In the 1975/76 survey only 0.5% of the respondents didn't know its sex. In this survey this figure had increased to 2.5%. These figures have not been tested for statistical significance!)

Table 2 gives the proportion of men and women in various job descriptions in data processing.

2.1.3 Time in Data Processing

Half of the respondents have spent less than 6½ years in data processing. The length of time respondents have been in data processing is shown in Figure 2.

Figure 2: Years in Data Processing



In Figure 3 is shown the distribution of the age of respondents when they entered the data processing profession:

FIGURE 3: AGE ON ENTERING DATA PROCESSING

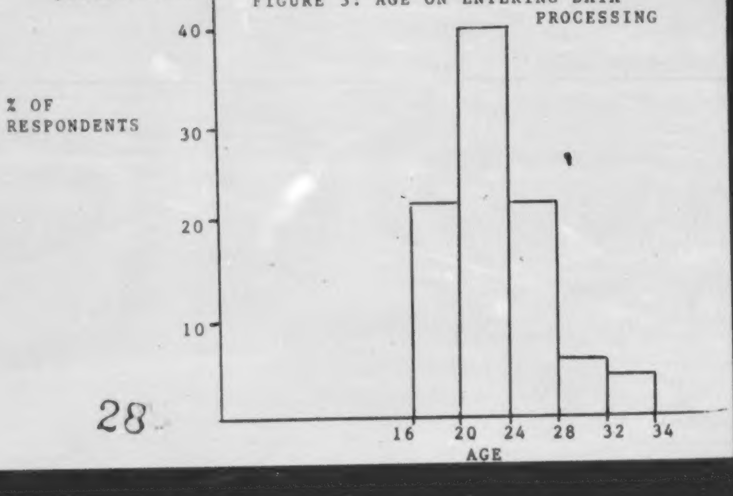


Figure 3 shows that while some (31%) people begin their career in data processing very young (<21) many are relatively "mature" when they start. This implies the latter are gaining experience (either in a work environment or at a tertiary educational institution) before starting in data processing.

2.1.4 Job Distribution within Data Processing

Table 1 gives the percentage of respondents occupying specific occupational classifications.

Table 1: Job Distribution in Data Processing

POSITION	AGE RANGE	% IN THAT POSITION		
		Men	Women	Total
Trainee Programmer	18-29	2	2	4
Assistant Programmer	19-27	3	2	5
Applications Programmer	20-32	9	4	13
Senior Applications Programmer	22-42	3	3	6
Supervisory Applications Programmer	22-44	3	2	5
Chief Programmer	24-36	2	1	3
Systems Programmer	21-34	5	-	5
Junior Systems Analyst/Designer	21-25	1	1	2
Analyst/Programmer/Systems Designer	22-39	12	2	14
Senior Systems Designer	23-44	7	-	7
Chief Systems Designer	26-44	2	-	2
Systems Development Manager	28-54	4	-	4
Data Processing Manager	24-56	-	-	15
Education/Training/Support	25-52	5	1	6
Other	-	-	-	9

From Table 1 it is apparent that 36% of those sampled are either trainee programmers or some category of applications programmer, 25% are in systems analysis and design, and 19% are involved in management.

2.1.5 Previous Experience

Respondents were asked to indicate what they had been doing immediately prior to beginning their career in data processing. Their replies are given in Table 2.

Table 2: Immediate Previous Experience.

CATEGORY	PERCENTAGE
Clerical work	37
Non-clerical work	19
Full time education	42

There is a noticeable trend for the more recent entrants (younger than 30) to come direct from full time education than has been the case in the past. The proportion of women entering data processing directly from full time education was 56%.

2.2 Educational Qualifications

Respondents were asked to indicate all educational qualifications they had obtained or anticpated obtaining within the next three years. The results are summarised in Tables 3 and 4.

Table 3: Educational Attainment.

QUALIFICATION	PERCENT	
	HAVE IT	ANTICIPATED WITHIN NEXT 3 YEARS
School Certificate	81	0
University Entrance	74	0
Partial Batchelors Degree	19	1
Batchelor's Degree	27	3
Masterate or Honours Degree	11	1
Ph.D.	2	1
Partial DP Certificate	7	1
NZ Certificate in DP	4	4
Other Technicians Certificate	3	0
Post Graduate Diploma in DP	2	0
Undergraduate Diploma	1	0

Table 4: Highest Educational Qualification

TERTIARY QUALIFICATION	PERCENTAGE
None	58%
Undergraduate Diploma	10%
Degree	32%

It is clear from Tables 3 and 4 that the average person involved in data processing is not particularly well-qualified. In order to see whether this is changing as the data processing profession matures, figures were taken out on the relative proportions of those with various qualifications as a function of their age and the time they have been involved with data processing.

Approximately 35% of those under 24 had a cetificate in data processing. This is much higher than the corresponding figure for those 24 or over, not surprising in that diplomas have only been available over the last 4-6 years. There is a slight trend for there to be proportionately more graduates in the 24-35 age group than in the older members of the profession.

There was no difference between men and women in their level of educational qualification.

2.2 Professional Activities

The proportion of respondents belonging to various bodies is given in Table 5. In this table the pooled numbers of members of all grades belonging to each society is also given.

Table 5 : Membership of Professional Bodies.

SOCIETY/PROFESSIONAL BODY AND GRADE	PERCENTAGE OF RESPONDENTS	TOTAL PERCENTAGE FOR SOCIETY
New Zealand Computer Society		
Associate	27.6	
Member	8.5	
Fellow	0.5	37.6
British Computer Society		
Associate	0.5	
Member	1.9	2.4
Association for Computer Machinery		
Member	0.5	0.5

Chartered Institute of Secretaries		
Associate	0.4	
Member	0.5	0.9
New Zealand Society of Accountants		
Member	4.2	4.2
New Zealand Institute of Management		
Associate	8.3	
Member	0.8	9.1
Other	0.5	0.5

In total 43% of respondents belonged to a professional body at some level while 40% belonged to a computer society.

2.4 Income

Respondents were asked to give their income in three categories (a) their annual salary, (b) perquisites (such as annual value of car, car allowance, profit sharing, superannuation, etc) and (c) supplementary income earned from computing (programming, etc.). The responses to these is summarised in Tables 6 and 7 and discussed in detail below. It should be remembered that these figures refer to the period December 1977 to February 1978.

2.4.1 Salary

The overall average salary was $10,000 with the average being $10,600 for men and $7,700 for women. In general those who reported that they found their job satisfying had a considerably higher average salary than the others. It is not clear whether the high salary gave the high job satisfaction or vice versa. Salary and total income levels were independent of the size of the data processing section for which the respondent worked. Those who anticipated that their next job would not be in data processing had a higher salary than the others. This is most probably related to the fact that many of these high salaries would have been earned by management staff who would have to leave data processing for career progression. Those belonging to the New Zealand Computer Society had a higher salary and those only having a data processing certificate had a lower salary than the remainder. Both of these effects are a consequence of the relative age and youth of the above two groups. In general, the average salary rises until age 48 (Figure 4)

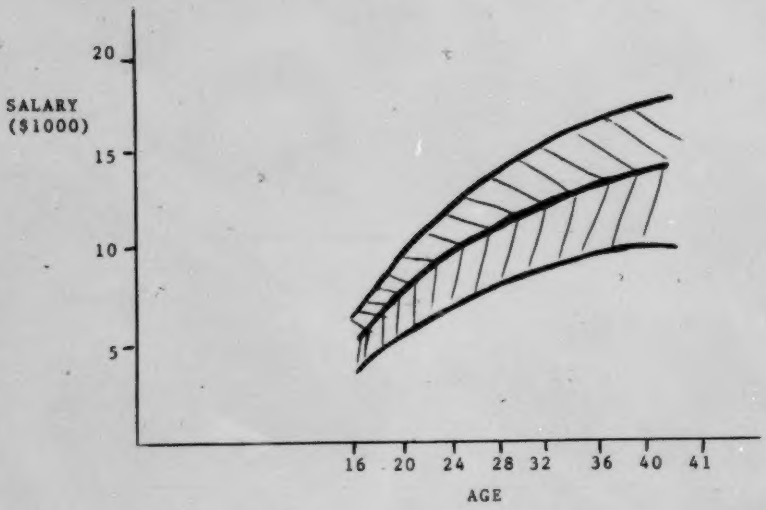
Table 6: Annual Salary and Total Income Distribution.

RANGE ($)	SALARY (%)	TOTAL INCOME (%)
5000- 6000	5	3
6000- 7000	7	7
7000- 8000	10	10
8000- 9000	11	10
9000-10000	11	11
10000-11000	16	13
11000-12000	12	11
12000-13000	9	7
13000-14000	7	7
14000-15000	3	3
15000-16000	4	5

Table 7: Average Salary and Ranges for Specific Functions

	AVERAGE	RANGE
Trainee Programmer	6,700	5000- 8000
Assistant Programmer	6,500	5000-10000
Applications Programmer	7,400	5000-11000
Senior Applications Programmer	8,700	6000-12000
Supervisory " "	9,300	7000-12000
Chief Programmer	11,100	9000-15000
Systems Programmer	9,100	6000-15000
Junior Systems Analyst/Designer	8,400	6000-12000
Analyst/Programmer/Systems "	9,400	7000-14000
Senior Systems Designer	11,000	8000-15000
Chief Systems Designer	11,500	10000-15000
Data Processing Manager	13,800	9000-20000
Education/Training/Support	11,400	7000-19000

FIGURE 4: SALARY RANGE AS A FUNCTION OF AGE



2.4.2 Perquisites

The results from the analysis of perquisites are a little puzzling in that 60% claimed to get no perquisites. Either respondents forgot to include pension scheme supplements from their employers or a considerable number are not getting them. The results are given in Table 8

Table 8 : Annual Value of Perquisites.

VALUE ($)	% OF RESPONDENTS
Nil	60
<1000	19
1000-2000	9
2000-3000	6
3000-4000	4

There is a significant sex distinction in the awarding of perquisites. While 25% of the men get more then $1000 in perquisites, only 6% of the women do. This discrimination is most probably due to job distribution where proportionately more men attain higher paid positions than women. Perquisites are generally associated with more senior positions, for example 60% of data processing managers (a principally male area) get an average perquisite of $2,500 per annum. In general, if an employee is to receive any substantial level of perquisites he will have spent approximately 5 years with his present employer and/or have been in the same job description more than three years and/or be over 28. The awarding of perquisties was independent of the size of the firm.

2.4.3 Supplementary Income

Almost all (94%) respondents indicated that they earned no supplementary income from computing. Of those who did extra computing, most (4% of all replying) earned less than $1000 and 2% of the total earned between $1000 and $2000. There were only five people in the sample who were earning more than $2000. The biggest amount of supplementary income earned from computing was $11000.

2.4.4 Total Income

The figures supplied by each respondent for salary, perquisites and supplementary income were aggregated and the results are shown in Table 6. The average annual total income was $10,800 with men averaging $11,400 and women $7,900. This major gap between salaries for men and women is not all attributable to sexism but is a function of a number of effects, principally that women don't reach the higher salaried positions. This will be discussed further in the next section.

2.4.5 Calculate your(1977-78) Salary!

In the report we presented two years ago (1) we gave a number of graphs showing salary as a function of age, years in data processing, job description and sex.

We have used the information obtained in this survey to try to predict an individual's salary given sufficient details about him/her We have done this using the technique of stepwise linear regression analysis. This assumes a linear relationship between each of the factors (e.g. age) determining the salary and the actual salary, an assumption that may not be justifiable. However, the results are of sufficient interest to present here. For the benefit of the non-statistical the procedure for calculating an individual's salary is given as a set of instructions in Table 9. It must be stressed that any individual's actual and predicted salary may not be the same. However, the salary as calculated will approximate the average salary obtained by a person with those characteristics. It should also be remembered that these figures relate to December 1977-February 1978. Inflation will have rendered the data obsolete for present conditions.

The material in Table 9 has been ordered for convenience and does not indicate the importance, or weight, to be attached to each variable in determining its contribution to the total salary. The five variables which are the most important as far as determining salary is concerned are (in decreasing order): the number of years in data processing, whether the individual is a data processing manager, the age, the sex and whether the employer is a consultant. Of these, the only one which negatively influences the salary is if the person is female. There thus seems to be definite evidence that women are not paid at the same rate as men even when all other things are equal.

Table 9: Salary Calculation

		$
Base Salary		6070
For every year involved in data processing, ADD		180
For every year with present employer, ADD		65
For every year in age, ADD		60
If a woman, SUBTRACT		690
If possessing a certificate in data processing	SUBTRACT	460
If possessing a Batchelor's degree, ADD		510
If possessing a Master's or Honours degree, ADD		570
If possessing a Ph.D., ADD		790
If job description is (choose one of):		
Trainee Programmer	SUBTRACT	710
Assistant Programmer	SUBTRACT	1160
Applications Programmer	SUBTRACT	480
Chief Programmer	ADD	1520
Analyst/Programmer/Systems Designer	SUBTRACT	100
Senior Systems Designer	ADD	1120
Chief Systems Designer	ADD	1520
Systems Development Manager	ADD	2560
Data Processing Manager	ADD	2873
If the employer is (choose one of):		
Government	SUBTRACT	440
Education	SUBTRACT	830
Computer Bureau	ADD	380
Consultant	ADD	1010

2.5 Job Satisfaction

2.5.1 Desirable Job Attributes

Respondents were asked to indicate, from a supplied list, what they considered were the two factors (a) which made their present job more satisfying and (b) which they would find most satisfying in any job. The results are given in Table 10.

The relative rank of these attributes was independent of the level of job satisfaction (see 2.5.2) and the size of the DP department they worked for. There were some slight differences in attitude among those working for different employers. For example, those working for bodies engaged in education wanted a high technical challenge but were not finding it. They also were more interested in a high salary than their other colleagures. This latter perhaps could be attributed to the fact that they were earning less than their peers in the industry. As well, those working in bureaux and for consulting firms would like but do not get good working conditions. However, they do not seem to be ready to leave the data processing industry as a consequence.

Table 10: Desirable Attributes of a Job.

ATTRIBUTE	FOUND IN PRESENT JOB	% OF RESPONDENTS DESIRED IN ANY JOB	BOTH DESIRED IN ANY JOB AND FOUND IN PRESENT JOB
High Technical challenge	28	38	21
Using modern equipment	13	8	4
High Salary	11	25	6
Good Working Conditions	30	30	20
High Calibre Colleagues	22	20	13
Encouragement of initiative and responsibility	67	47	40
Flexible working Hours	8	18	5

It is very interesting to note that the most important attribute is the encouragement of initiative and responsibility and that many respondents are finding it in their present job. The next most desirable is high technical challenge while a high salary is comparatively unimportant. These are confirmed in an article on staffing quoting an Australian computing personnel consultant (2).

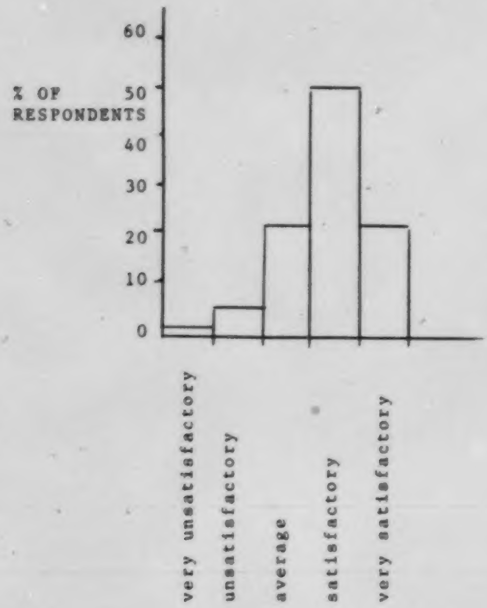
"Interesting work and progressive attitudes in a company are more likely than money to attract good computer staff"

Women did not consider high technical challenge as important as men and considered flexible working hours more important.

2.5.2 Job Satisfaction

Respondents were asked to grade their level of satisfaction with their present position. Results are shown in Figure 5:

Figure 5: Job Satisfaction



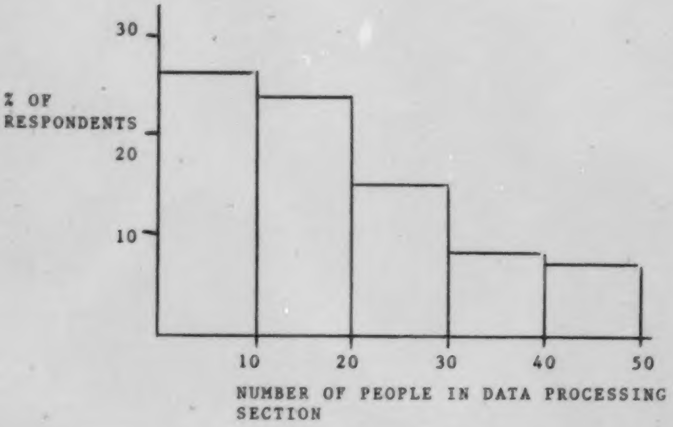
This was generally independent of sex, job description or age.

2.6 Employment Characteristics

2.6.1 Size of DP Section

Respondents were asked to indicate the total number of staff in their organisation who were directly involved in data processing. This included operators, data preparation staff, programmers, systems analysts, data processing managers, etc. Figure 6 shows the resulting distribution. Less than 5% of respondents were self employed.

Figure 6: Size of Data Processing Establishment



Not unexpectedly, 50% of the respondents worked for firms with a data processing group of less than 20 people.

2.6.2 Employer Category

The proportion of respondents working for different categories of employers is given in Table 11. Of interest is that the relative numbers of people working in government and private bureaux are approximately the same. Almost half of the respondents work for non-computing specialist firms, i.e. commercial firms with a data processing department.

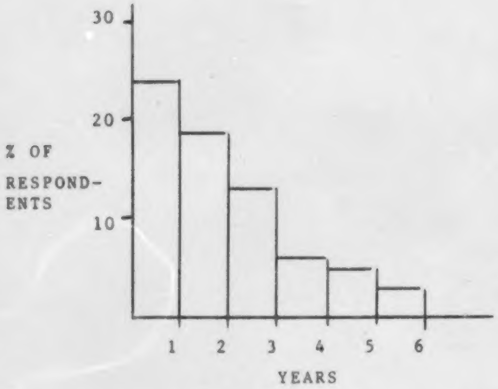
Table 11: Respondents Categorised by Type of Employer

TYPE OF EMPLOYER	% EMPLOYED
Government	22
Education	7
Computer Bureau	19
Consulting	9
Other Commerce	43

2.6.3 Time with Current Employer

As well as being asked to indicate the length of time they had held specific positions within data processing, respondents were asked to show how long they had worked for their current employer. The results are shown in Figure 7. Half those returning questionnaires had worked for their present employer for 26 months or less!

Figure 7: Time spent with Present Employer.



Because of some very long serving staff, the average length of time spent working for the current employer was 51 months. However, women had an average much lower than this (29 months). This could be due to the fact that women are much more recent entrants to data processing (and other professions) than men so that there are no long servers to bring up the average for women. This is supported by the fact that the average length of time in the current job description is the same for men and women. The average age for men (30) is also greater than for women (24).

2.7 Career Expectations

Most (87%) of the respondents expected that their next job would be within data processing. This expectation was independent of sex, or the length of time in data processing. It was also, generally, independent of the respondent's job description except for data processing managers, 30% of whom expected that their next job would not be in data processing.

2.8 Career Development

2.8.1 Career Paths

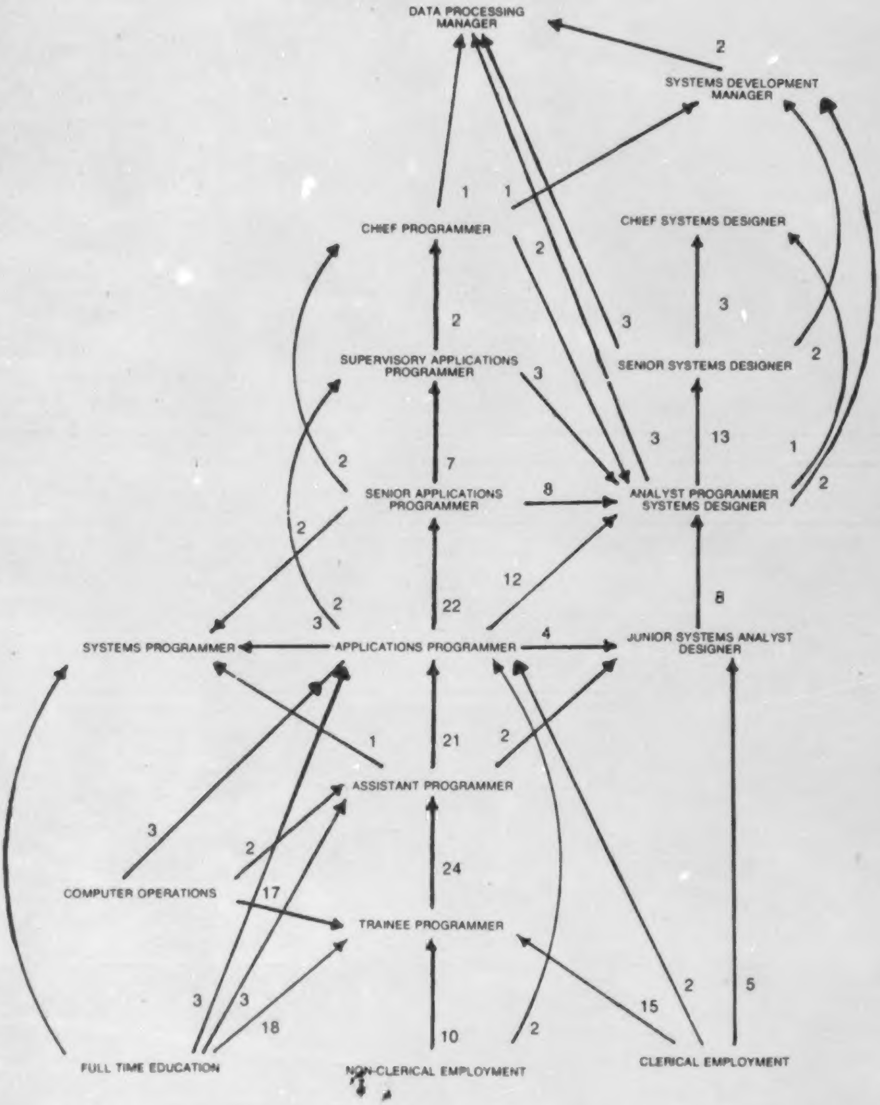
Using the career paths traced out by respondents to the questionnaire it has been possible to construct an overall perspective of the career progression of people employed in data processing. This broad overview is shown schematically in Figure 8. In this migration from position to position is shown by an arrow linking two job titles. The number associated with the arrow is the approximate percentage of the total number of people entering data processing who take that path. Thus, of the total number of people involved in computing 18% went from full time education to become a trainee programmer. Similarly 24% of those who began a career in computing went from the trainee programmer position to become assistant programmers. The descriptions of these job titles are given in the questionnaire (Appendix 1). In order to preserve some level of clarity only major (generally ⟩ 2%) paths are shown.

In our first study we were criticised for making no provision for people occupying the same job description more than once. The present questionnaire was designed to avoid this problem by providing extra "boxes" for the respondent to fill in appropriately. Analysis of the responses showed that less than 5% of those surveyed occupied the same position twice or more. Also, few (3%) of those surveyed left data processing careers for different employment and then returned. Very few people reported "demotion" moving from a position to a more junior one within the same area. Similarly, few data processing managers reported moving to other positions within data processing.

Examination of Figure 8 shows that promotion to data processing manager is by way of the systems area rather than via programming. Few senior programmers become data processing managers without first migrating into systems analysis.

One aspect of the analysis of career paths which is of particular interest is in the recruitment of systems analysts. Discussion with DP managers suggests that the ideal recruit to systems analysis is one with a good knowledge of the firm. Programming skill is considered relatively unimportant. The reality is quite different. Only 20% of all those who have been junior systems analysts/designers have come from full time clerical employment. Almost all of the remainder of people who become systems analysts were recruited from the programmer categories.

FIGURE 8: CAREER PATHS IN DATA PROCESSING


DATA PROCESSING MANAGER
2
SYSTEMS DEVELOPMENT MANAGER
1
1
CHIEF PROGRAMMER
CHIEF SYSTEMS DESIGNER
2
3
3
2
2
SUPERVISORY APPLICATIONS PROGRAMMER
3
SENIOR SYSTEMS DESIGNER
2
3
13
1
2
7
SENIOR APPLICATIONS PROGRAMMER
8
ANALYST PROGRAMMER SYSTEMS DESIGNER
2
2
2
12
22
8
3
SYSTEMS PROGRAMMER
APPLICATIONS PROGRAMMER
4
JUNIOR SYSTEMS ANALYST DESIGNER
1
21
2
ASSISTANT PROGRAMMER
3
2
24
COMPUTER OPERATIONS
17
TRAINEE PROGRAMMER
3
3
18
10
2
15
2
5
FULL TIME EDUCATION
NON-CLERICAL EMPLOYMENT
CLERICAL EMPLOYMENT

2.8.2 Job Occupancy Times

The range and average length of time respondents spent in various positions in data processing is given in Table 12.

Table 12: Total Time in Jobs.

POSITION	TIME IN POSITION (MTHS)	
	Average	Range
Trainee Programmer	7	2-18
Assistant Programmer	9	3-24
Applications Programmer	18	2-54
Senior Application Programmer	20	3-30
Chief Programmer	20	3-60
Systems Programmer	26	6-66
Junior Systems Analyst/Designer	15	3-41
Analyst/Origrammer/Systems Designer	25	6-66
Senior Systems Designer	28	2-66
Chief Systems Designer	23	5-50
Systems Development Manager	34	2-72
Data Processing Manager	45	5-132

3. GENERALISATION OF RESULTS

3.1 Comparison with 1975/76 Survey

In 1976 the authors reported on a pilot survey of New Zealand data processing staff (1). There is a considerable commonality between the questionnaire used then and that used for the survey reported here. It is thus possible to compare the two surveys for consistency. Agreement between them, given the completely different sampling method (20 selected firms vs. blanket approach) and sample size (200 vs. 875) will indicate that the results can be generalised to the New Zealand data processing profession as a whole.

In fact there is considerable agreement between the two surveys. The age distribution, number of years in data processing, previous experience, educational attainment and job satisfaction all exhibit substantial agreement in both surveys.

3.2 Comparison with a Management Survey

In a separate report (3) the authors have analysed responses of 20 data processing managers in the lower half of the North Island to a survey of current staff and future requirements. The ratio of staff having degrees, data processing certificates and no qualifications were compared with that found in this survey. The ratios were virtually identical

3.3 Comparison with NZCS Statistics

Excluding student members, the proportion of Associates to Members to Fellows within the New Zealand Computer Society was 77:23:0.5. The corresponding ratio for those returning the questionnaire who were members of the NZCS was 73:25:1. This is an extremely close agreement, suggesting that this survey and the NZCS membership are sampling from the same population.

3.4 Generalisation

On the basis of the close agreement with the above three comparisons it is justifiable to extend the results reported in this paper to the New Zealand data processing community as a whole. Thus all the statistics reported in Section 2 can be regarded as typical of computing personnel in New Zealand.

4. DISCUSSION

4.1 Job Occupancy

In spite of the agreement between the 1975/76 and the 1977/78 surveys any attempts to detect trends over the intervening period would be dangerous. One area in which there is a strong discrepancy is the length of time respondents reported holding various job descriptions. The authors suggested in 1976 that the very short times people claimed to occupy various positions was such as to raise doubts about the description of data processing as a "profession". The job occupancy times reported here are considerably longer than in 1976. This is most probably due to the fact that in the present survey the jobs were specified closely. People could not indulge in job inflation as may have happened in the earlier survey when specifications were not provided. It is still somewhat disturbing to note that 50% of those surveyed changed their employer within the last 26 months. This is a very high job mobility.

4.2 Salary Movement

One other comparison between the 1975/76 and 1977/78 surveys that can be made with comparative safety is between the average salaries reported in both surveys. In 1976, the average salary, over all respondents was $7,700. In 1978 it was $10,000. This corresponds to a 12.5% per annum increase in salary scales overall. The increase in mens' salaries over this period was from $7,900 to $10,600, while for women it was from $6,100 to $7,700. These increases are proportionally the same as for the overal increase.

4.3 Data Processing Numbers in New Zealand

The excellent agreement between the Computer Society Membership figures and those obtained in this survey makes it reasonable to attempt to predict the number of people involved in those areas covered by this questionnaire. NZCS members comprised 37.6% of respondents. Given that the NZCS had 1545 non-student members in 1977 the number of people involved in programming, systems analysis and management in New Zealand is therefore close to 4100.

Given an average annual salary of c.$10,000 this means that the total salary bill for computing professionals in New Zealand is of the order of $40,000,000.

ACKNOWLEDGEMENTS

The authors would like to thank: The New Zealand Computer Society and the Massey University Humanities and Social Science Research Committee for grants to cover costs associated with this work reported here.

The Wellington IBM Users Group for use of their job descriptions.

The Management and staff of those firms who co-operated by completing and returning the questionnaires.

REFERENCES

(1) C.R. Boswell and P.J. Melhuish
"Changes in the Methodology of Programming - A new look at the Task of Applications Programming in Data Processing"
Proceedings of the Fifth New Zealand Conference of the New Zealand Computer Society 1976 pp. 215-248.

(2) Pacific Computer Weekly, May 12th 1978, p.20

(3) P.J. Melhuish and C.R. Boswell
"Computer Education in New Zealand - the Dream and the Reality"
Proceedings of the Sixth New Zealand Conference of the New Zealand Computer Society, 1978

APPENDIX 1

Massey
University

the New Zealand
Computer Society

Victoria
University of
Wellington

EDP CAREERS SURVEY - NEW ZEALAND 1977

INTRODUCTION

We are currently investigating the career and salary structure of EDP professionals. At this stage we are specifically interested in the career path and expectations of personnel employed in the areas of programming, systems analysis and EDP management. This survey will enable us to gather some of this information.

This questionnaire is in two sections. Using the first section we will determine some basic characteristics of people entering the field of EDP. The second part, by the use of a simple but cunning diagram, will analyse the various stemps taken by them, to reach their present position. In addition, we will build up a profile of salary ranges for the various EDP occupations

We very much appreciate your co-operation and assure you that individual responses will be treated with complete confidentiality.

P.J. Methuish
Department of Computer Science
Massey University
Palmerston North

C.R. Boswell
Computing Services Centre
Victoria University of Wellington
Wellington

APPENDIX 1 (Cont'd....)

PART ONE

PERSONAL DETAILS

Year of birth	Y ☐ Y ☐ 1/ 7 – 8	Sex	Male ☐ Female ☐ 9 – 10	Date that you entered EPD	M ☐ M ☐ Y ☐ Y ☐ 11 12 13 14

EDUCATIONAL QUALIFICATIONS

Please tick the appropriate boxes in the table below.

Qualification	Obtained	Anticipated obtaining within the next three years
School Certificate	☐ 15	☐ 16
Sixth Form Certificate	☐ 17	☐ 18
University Entrance	☐ 19	☐ 20
Partial Bachelor's Degree	☐ 21	☐ 22
Bachelor's Degree	☐ 23	☐ 24
Masters or Honours Degree	☐ 25	☐ 26
Ph.D.	☐ 27	☐ 28
Partial DP Certificate	☐ 29	☐ 30
New Zealand Certificate in DP	☐ 31	☐ 32
Post Graduate Qualification in DP *(Please specify)*		
....	☐ 33	☐ 34
....	☐ 35	☐ 36
Other *(Please specify)*		
....	☐ 37	☐ 38
....	☐ 39	☐ 40

MEMBERSHIP OF PROFESSIONAL BODIES

Please tick boxes corresponding to any bodies to which you belong.

Associate New Zealand Computer Society	☐ 41	Associate Chartered Institute of Secretaries	☐ 42
Member New Zealand Computer	☐ 43	Member Chartered Institute of Secretaries	☐ 44
Fellow New Zealand Computer Society	☐ 45	Member New Zealand Society of Accountants	☐ 46
Associate British Computer Society	☐ 47	Other Professional Bodies *(Please specify)*	
Member British Computer Society	☐ 48	....	☐ 49
Member of Assn. of Computing Manufacturers	☐ 50	Other Professional or Technical qualification *(Please specify)*	
Other Computer Societies *(Please specify)*		....	☐ 51
....	☐ 52	....	☐ 53
....	☐ 54	....	☐ 55
....	☐ 56		

APPENDIX 1 (Cont'd....)

INCOME

Please enter (to the nearest $100) your current annual salary. $ ☐☐☐ 0 0 57 59

Please enter (to the nearest $100) the approximate annual value of perquisites (car, car allowance, profit sharing, superannuation, etc.). $ ☐☐☐ 0 0 60 62

Please enter (to the nearest $100) the value of any supplementary income you earn from computing (programming, etc.). $ ☐☐☐ 0 0 63 65

JOB SATISFACTION

Please indicate (by ticking the appropriate boxes) the two most important factors which you consider would make any job you held satisfying.		*Please indicate (by ticking the appropriate boxes) up to two characteristics in your present job which gives you the most satisfaction.*	
High technical challenge	☐ 66	High technical challenge	☐ 67
Using modern equipment	☐ 68	Using modern equipment	☐ 69
High salary	☐ 70	High salary	☐ 71
Good working conditions	☐ 72	Good working conditions	☐ 73
High calibre colleagues	☐ 74	High calibre colleagues	☐ 75
Encouragement of initiative and responsibility	☐ 76	Encouragement of initiative and responsibility	☐ 77
Flexible working hours	☐ 78	Flexible working hours	☐ 79

Please indicate (by ticking the appropriate box) how satisfying you find your present position.

VERY UNSATISFYING	UNSATISFYING	AVERAGE	SATISFYING	VERY SATISFYING

2/7

EMPLOYMENT TYPE

Please enter the approximate total number of staff involved in the EDP part of the organisation you work for (This will include operators, data preparation staff, programmers, systems analysts, DP management).

☐☐☐ 8 10

Please indicate the category which most accurately describes the organisation for which you work.

Government	☐ 11
Education	☐ 12
Computer Bureau	☐ 13
Consulting	☐ 14
Other Commerce	☐ 15

Please indicate the length of time in months you have worked for your current employer. M M M ☐☐☐ 16 18

CAREER EXPECTATIONS

Do you think that your next job will be within EDP? *(Please tick)* YES ☐ NO ☐ 19 20

APPENDIX 1 (Cont'd....)

PART TWO

CAREER DEVELOPMENT IN EDP

On the next page, we have indicated the major occupational classifications of people engaged in EDP. Each box contains a job title, and space for you to enter the length of time that you may have spent doing that job.

Please attack the form in this way:

(i) Starting at one of the given "starter" boxes at the bottom of the page, draw a line to the first position that you held in EDP.

(ii) Enter in that box, the length of time in years and months that you spent doing that job.

(iii) If that job is NOT your current job, draw a line to the next position that you held.

(iv) Repeat steps (ii) and (iii) until you do reach your current job. Enter, in your current position, the length of time that you have been employed in that job. If you have held the same position more than once, indicate the position and time for the second and later occupencies in the spare boxes on the right.

PLEASE NOTE the following points:-

(i) Job descriptions are notoriously vague and subjective. To help you we have attached, at the back of the questionnaire, descriptions of the qualifications and responsibilities corresponding to each job title. Please select that description which you think most accurately describes your function. Only if nothing in the list comes anywhere near describing what you do then use one of the spare boxes.

(ii) If you changed employer, but went on doing the same job, do not try to show that on the form. We are only interested in the TOTAL TIME that you spent employed in a particular job category.

(iii) If you divide your time between two or more job titles (e.g. as a Systems Designer AND Applications Programmer) indicate that job in which you spent a majority of your time.

APPENDIX 1 (Cont'd....)

EDP CAREERS SURVEY NEW ZEALAND 1977

PART TWO – CAREER DEVELOPMENT

17

SYSTEMS-DEVELOPMENT MANAGER			
Y	Y	M	M

16

DATA PROCESSING MANAGER			
Y	Y	M	M

14

CHIEF PROGRAMMER			
Y	Y	M	M

12

SUPERVISORY APPLICATIONS PROGRAMMER			
Y	Y	M	M

10

SENIOR APPLICATIONS PROGRAMMER			
Y	Y	M	M

7

SYSTEMS PROGRAMMER			
Y	Y	M	M

8

APPLICATIONS PROGRAMMER			
Y	Y	M	M

4

COMPUTER OPERATIONS			
Y	Y	M	M

5

TRAINEE PROGRAMMER			
Y	Y	M	M

STARTER BOXES

1 CLERICAL EMPLOYMENT

2 NON-CLERICAL EMPLOYMENT

377

APPENDIX 1 (Cont'd....)

OTHER SENIOR EDP POSITION *(Please specify)*

Y	Y	M	M

Y	Y	M	M

18

CHIEF SYSTEMS DESIGNER

Y	Y	M	M

Y	Y	M	M

12

SENIOR SYSTEMS DESIGNER

Y	Y	M	M

Y	Y	M	M

SPARE UN-NAMED BOXES IF YOU NEED THEM

11

ANALYST/PROGRAMMER SYSTEMS DESIGNER

Y	Y	M	M

Y	Y	M	M

9

JUNIOR SYSTEMS ANALYST DESIGNER

Y	Y	M	M

Y	Y	M	M

8

ASSISTANT PROGRAMMER

Y	Y	M	M

Y	Y	M	M

FULL TIME EDUCATION

APPENDIX 1 (Cont'd....)

6

TITLE	QUALIFICATIONS	TYPICAL RESPONSIBILITIES
Trainee Programmer	Minimum pass in aptitude test, experience in operations, attended a University or Polytech course or has company experience. Age range flexible but unlikely to be aged over 20 years.	Works from completed detailed specifications. Requires continual supervision in early stages, to lesser degree as training period reaches conclusion.
Assistant Programmer	Has satisfactorily completed training period as a Trainee Programmer. Should be capable of preparing straightforward computer programs.	Programs from detailed specifications. Requires regular direct supervision but can show competence in straightforward programming operations.
Applications Programmer	Satisfactory completion of Assistant Programmer period or equivalent. Requires some corporate knowledge.	Programs all but exceptionally complex programs. Works from broad specifications. Requries little supervision. Able to think through to the finished job and to maintain programs written by other programmers.
Senior Applications Programmer	Requires sound coporate knowledge. Must have substantial programming expertise.	Handles any application program or package. Does not require detailed specifications. Maintains programs written by other programmers. Participates in system development and program specifications. Renders assistance to lower levels of programmers. Shows initiative and maturity.
Supervisory Applications Programmer	Requires corporate knowledge. Has substantial programming experience. Must be capable of organisation and supervision of work of other programmers.	Handles any application program or package. Does not require detailed specification. Maintains programs written by other programmers. Participates in system development and program specification. May supervise work of others. Plays major part in implementation and maintenance.
Chief Programmer	Requires corporate knowledge. Has substantial programming experience and has directed a number of major programming projects; must be capable of supervising other programmers, and be aware of overall objectives of all current programming projects.	Overall responsibility for standards of program design, implementation and testing throughout the installation; supervises training and develoment of programming staff; allocates programming resources; reports directly to DP management.
Systems Programmer	Has a minimum two years programming experience and expertise in all programming languages used in the installation. Possesses intimate knowledge of architecture of machine and operating systems. Has knowledge of technical aspect of installed applications software. Has knowledge of file organisation and accessing methods.	Generates and maintains the machine operating system. (This includes macro writing, utility writing, and other extensions to the software systems). Isolates hardware/software problems and analyses errors to determine whether in-house or machine company action is required. Normally consulted on planning and use of disk space. Keeps up to date with, and others informed of, hardware and software developments. Handles application or package programs where special expertise is required. Acts as a technical consultant to Operations and Development staff. May supervise the work of others.
Junior Systems Analyst/Designer	May have only marginal computer programming knowledge; has experience of Company (or related procedures). Above average ability in communication skills; able to demonstrate potential in areas of problem definition and solution.	Works under direct supervision of Analyst/Programmer/Systems Designer, or Senior Systems Designer, in areas of systems analysis and specification.

APPENDIX 1 (Cont'd....)

TITLE	QUALIFICATIONS	TYPICAL RESPONSIBILITIES
Analyst/Programmer Systems Designer	Is proficient in programming languages and design techniques in use in own installation. Has at least two years experience with operational systems. Must possess proven ability in defining problems and solutions. Has above average ability in oral and written skills and has ability to work with others.	Determines nature of problem and presents computer oriented solution. Defines resources required for development and operation of solution. Defines the system as it affects computer and user functions. (Computer aspects to be specified in sufficient detail to permit programming). May undertake programming. Maintains involvement until implementation completed.
Senior Systems Designer	Requires corporate knowledge and thorough knowledge of programming techniques. Has at least two years working association with computer based systems and a minimum of two years experience as a Systems Designer. Must possess proven ability in problem analysis, definition and solution. Has above average ability in oral and written communication. Must demonstrate ability to get on with people. Has ability to communicate with senior management. Operates with minimal supervision.	Determines nature of a problem and recommends computer oriented or alternative solutions. Defines resources required for development and operation of solution. Defines the system as it affects computer and user functions. (Computer aspects to be specified in sufficient detail to permit programming). Maintains involvement until implementation completed. Capable of producing solutions to larger or more complex problems (i.e. new systems or modifications to existing systems) requiring in excess of one man year development effort. Establishes implications of proposed solution on other systems.
Chief Systems Designer	Capable of organising, directing and controlling system development staff. Able to negotiate with senior management.	Assists with the development of short term (i.e. up to one year) work plans for his staff. Controls complete project development and allocated maintenance. Co-ordinates activities of all his staff with other sections in the computer department. Suggests system improvements and new schemes. Assists in appraisal and development of his subordinate staff.
Systems Development Manager	Extensive experience of organising, directing and controlling systems development staff; substantial record of implementation of computer based systems; essential that he be able to negotiate with all levels of (including very senior) management.	Complete supervision of Systems Development personnel in areas of selection, design and implementation of computer based systems.
Data Processing Manager	Has a thorough knowledge of all aspects of the DP department (although practical experience in all facets of DP work is not required); has detailed knowledge of the user requirements for DP, of budgeting and management techniques, and of hardware/software potential, both current and future.	Reporting directly to senior management, he is responsible for the efficient execution of all stages of the DP operation, including the design and development of computer based systems and the efficient running of these systems. He is directly responsible for hiring of staff and allocation of resources.

APPENDIX 1 (Cont'd....)

Finally,

Thank you for your co-operation.

If you would like to receive a brief summary of the results of this survey, write your name, and mailing address here:

____

____

____

If you have any comments regarding the survey, feel free to use this space.

Please send this questionnaire, as quickly as practicable to:

Colin Boswell
Computing Services Centre
Victoria University of Wellington
WELLINGTON

TEACHING OF COMPUTING IN NEW ZEALAND - THE DREAM AND THE REALITY

AUTHORS C.R. BOSWELL, M.Sc., Ph.D. (Massey) MNZCS
Director
Computing Services Centre
Victoria University of Wellington

P.J. MELHUISH, M.A. (Oxford), MBCS
Senior Lecturer
Department of Computer Science
Massey University

ABSTRACT

In 1977, under the joint auspices of the N.Z.C.S., Massey University and Victoria University of Wellington, the authors surveyed the depth and extent of courses in computing (in its widest sense) throughout New Zealand. Some of the results of that survey are presented.

Results of a random survey of data processing managers are given. In this survey, a number of data processing managers were asked to state the knowledge and skills that they would expect to find in applicants for "systems" or "programmer" vacancies. The data processing managers were also asked to indicate the relevance of a number of topics that could be included in courses in computing or data processing.

Comparisons are drawn between the three main elements in the education and training complex:

> The immediate skills required of the employee, as seen by the employer:
>
> The educational requirements of students and the requirements (both long and short term) of the industry, as perceived by the teaching institutions:
>
> The wishes of the student to reach an acceptably high level of work to justify a certificate, diploma or degree, and also to receive a useful "meal ticket".

Some possible new directions and emphases of computing teaching are also suggested in the hope of improving the level of communication between the providers of computer tuition, and the buyers of their products.

KEYWORDS

Computing courses, Training of Computer personnel, Education of Computer personnel, Pre-requisites for employment in DP

C.R. CATEGORIES

1.52, 1.9, 2.2, 2.42, 2.9, 3.51.

1. COMPUTING EDUCATION AVAILABLE IN NEW ZEALAND

1.1 A Survey of Courses Available

During the winter of 1977, a survey of the available courses in computing and data processing throughout New Zealand was undertaken. The aim of the survey was to centralize information regarding the type and extent of such courses, providing any interested enquirers with one standard source of data. Support and funding of the survey came from the Universities (both Massey and Victoria) and from the New Zealand Computer Society.

Particulars were obtained from relevant departments in the six universities in New Zealand together with Lincoln College, polytechnics and other technical college institutions, correspondence schools and computer manufacturers represented in New Zealand.

A complete list of the institutions and the departments or faculties that responded to the survey is given in Table 1.

TABLE 1: A LIST OF RESPONDENTS TO THE SURVEY OF COMPUTING EDUCATION IN NEW ZEALAND *

University of Auckland	Accountancy
	Board of Computer Studies
	Management Studies
	Mathematics
	Theoretical and Applied Mechanics
University of Canterbury	Accountancy
	Geography
	Mathematics
	Physics
	Computer Science
Lincoln College	
Massey University	Computer Science
	Business Studies
University of Otago	Science
	Medicine
	Quantitative and Computer Studies
	Commerce
Victoria University of Wellington	Information Science
	Economics
	Accountancy
University of Waikato	Computer Science
	Physics
Auckland Technical Institute	Mathematics and Data Processing
Central Institute of Technology (Lower Hutt)	Management
Christchurch Technical Institute	Applied Science
Otago Polytechnic	Science and General
Waikato Technical Institute	Construction
Wellington Polytechnic	Mathematics and Science
International Correspondence School	
Burroughs	
Control Data	
IBM	
NCR	

* Departments or establishments not listed in this table indicates no or null response, not that they were not asked for information.

The questionnaire is included in Appendix 1. Respondents were asked to give, for every course, information on:

course title and content

prerequisites for entry to the course

any qualification to which the course contributed

costs

teaching methods employed

number of students enrolled in 1977

This data has been stored in a machine readable file and from this file a comprehensive report has been produced. It is intended that this file be kept up-to-date so that it reflects changes in course structure, in material included in courses and in new courses offered. Of course, the effort to maintain this file, and to produce yearly (or preferably bi-yearly) reports is dependant upon the enthusiasm of both the data processing community at large, and also of the respondents to the survey. The mechanism for updating is clearly available. Full copies of the report are available from the NZCS. As an example of the output, two pages of the report are included in this paper, as Appendix 2.

1.2 Some Salient Features of the Survey

All universities in New Zealand and the principal technical colleagues are involved in computer education. The material taught by universities lies across the spectrum of computing from electronic design and software engineering through to business computing for accountants. The major emphasis in the technical institutes is on data processing training and education, generally within the framework of the New Zealand Certificate in Data Processing. The latter is administered by the Technicians Certification Authority. Many of these educational institutions provide special courses or seminars in aspects of computing, e.g. "Structured Programming in COBOL", "Data Communications" as NZCS Professional Development Seminars or "Introduction to Fortran Programming", just for interest. The major computer manufacturers represented in New Zealand support their clients in a variety of courses ranging from operator assistance, to advanced programming and systems techniques.

The variety and emphasis of courses offered reflects the pressures that are imposed on the various teaching bodies. On the one extreme, computer manufacturers provide (typically) short, intensive, training courses with a well-defined objective in terms of skills mastered or of situations appreciated. The subject material of the courses is chosen to correspond to the needs of the clients. Courses can be designed and implemented at short notice and can be tailor-made for any specific group of users.

Technical institutes typically teach within the framework of the New Zealand Certificate of Data Processing. Teaching methods vary, but the NZCDP papers are externally set, moderated, and marked. The system is fairly unresponsive to changes in the data processing environment. Students who qualify for the award of New Zealand Certificate in Data Processing have completed a substantial study unit, involving both theoretical and practical work, and covering a wide variety of data processing topics. Several technical institutes also teach towards a locally specified certificate in computing.

Universities, on the other hand, have a rather more flexible and independent situation than the technical institutes. New papers can be offered, provided that the university's authorities are convinced that:

The level of rigour and difficulty is appropriate

The practical work is adequately resourced

The teaching department can cope

It is not the role of the authorising bodies within universities to ensure that a particular paper is necessarily "relevant", attuned to the needs of the community, or any other such criteria.

University departments are therefore less affected by the pressing needs of the user than are the computer manufacturers. They are also substantially less disciplined in the choice of material they include than are the technical institutes. The result of this freedom is clearly indicated in the survey results, where different New Zealand universities demonstrate:

differing emphasis on computing

differing means of organising computing/data processing courses

differing levels to which data processing education is taken

In Table 2, some analysis of the courses offered across the range of "computing" is presented. Some of the points that arise from such an analysis are:-

The six technical institutes that responded to the survey all taught to the NZCDP syllabus;

The university courses invariably included some form of "introduction to computers and computing" course, often with high student numbers. In at least one university the first year course attracts (!) around 700 students.

Only three universities (Canterbury, Massey and Victoria) offered an Honours or Masterate degree in Computer Science.

TOPICS TAUGHT AT VARIOUS ESTABLISHMENTS

TOPIC	ORGANISATION																
	AUCK	CANT	MASS	OTAGO	VICT	WAIK	ATI	CIT	CHCH	OTAGO	WAIK	WGTN	CORR	BURR	CDC	IBM	NCR
GENERAL																	
Hardware & concepts	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
A programming language	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
COMMERCIALLY ORIENTED																	
Prog. - DP Elementary	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
DP Advanced			x				x	x	x					x		x	
Scientific	x	x	x	x	x	x	x		x				x			x	
Systems	x	x	x		x	x											
Systems Analysis & Design: Application to Accountancy																	
- Elementary	x	x	x	x	x	x	x	x	x	x	x	x					
- Advanced (e.g. MIS)	x	x			x	x											
Auditing and EDP		x					x	x	x	x	x	x					
Data Base Management Systems			x		x	x								x			
COMPUTER SCIENCE ORIENTED																	
Computer Science Theory																	
- Elementary	x	x	x	x	x	x											
- Advanced	x	x	x		x												
Computer Organisation & Architecture	x	x	x		x	x											
Algorithms & Data Structure	x	x	x	x	x	x	x	x	x	x	x	x					
Numerical Analysis		x	x	x		x	x		x								
Operation Research	x		x	x	x		x	x	x	x	x	x					
Operating Systems & Software Engineering	x	x	x	x	x	x	x	x	x	x	x	x		x		x	x
Data Communications Systems			x	x	x	x	x	x	x	x	x			x	x	x	
Modelling/Simulation	x	x	x	x	x	x									x	x	
Honours in Computer Science		x	x		x												

Most universities include a fairly strong, pragmatic, data processing oriented element in their computing education program. This strand is often taught outside the computer science department.

Differences in nomenclature between university departments would not help the layman:

> Massey, Canterbury and Waikato favour the term "Computer Science" as the departmental title.
>
> Victoria prefers "Information Science"
>
> Otago use the agglomerate "Quantitative and Computer Studies"
>
> Auckland have a "'Board' of Computer Studies"

While some universities still prefer to concentrate their computing staff in one department, in other cases the staff are members of the user faculties - e.g. Accounting, Engineering, Medicine.

Although generalisations are dangerous the survey produces substantial evidence of a width of computer-oriented education throughout the universities.

What is noticeable, however, is that the higher level papers (at third year bachelor's degree and beyond) are almost invariably in areas of "pure" or "technical" computer science.

Typical subjects are Compiler Construction, Programming Languages, Operating Systems, Artificial Intelligence and Machine Architecture. Only in fairly rare instances is there high level study of data processing oriented material, e.g. Data Base Management Systems, Management Information Systems, etc. A fairly strong interest in, for example, the role of the Auditor in EDP systems, as evidenced in the survey of Technical Institutes is not reflected in the universities responses. It is clear that the majority of the computing teaching effort in the universities is directed at either:

> The use of computer science as a general educational discipline (rather like Latin or Philosophy) providing computer science specialists for further computing research, and limited industrial employment.
>
> The provision of lower level "appreciation" courses, where much of the practical work is centred around programming and where there is little or no "Business Data Processing" emphasis.

There is little evidence of substantial enthusiasm for systems and business-oriented computing courses taken to high levels in graduate or post-graduate diplomas, honours courses or masterate programs.

2. DATA PROCESSING EDUCATIONAL REQUIREMENTS

A restricted survey of computer installations in the lower North Island was carried out in April and May of 1978. The questionnaire that was used, (Appendix 3) was designed to record the views of data processing managers on matters related to the recruiting and qualifications of staff. In particular, views were sought on the value of certain topics to be included in Computing/Data Processing courses. Experience in the background of applicants for entry into data processing departments was also investigated.

2.1 Method of Sampling and Administering the Questionnaire

Twenty organisations, located in Wellington, Palmerston North and Wanganui were arbitrarily selected. Government departments, computer manufacturer's bureaux, consultancy companies, commercial and manufacturing organisations were included. The size of the data processing departments in the programming, systems management areas, ranged from 3 to 30 people. The total number of employees in these categories for tnese 20 firms is shown in Table 3. The numbers in these categories is also expressed as percentages and compared with the equivalent values from a survey reported in another paper given at this Conference.(1). The questionnaire was administered in an interview situation with the data processing manager or the person in the department concerned with the hiring of staff.

TABLE 3: BREAKDOWN OF SAMPLE BY JOB TYPE

	Numbers in sample	Percentage in sample	Percentage in Larger Population (4)
Management	57	19	24
Systems	88	30	31
Programmers	150	51	45

2.2 Results of the Survey

2.2.1 Present and Future Staff Requirements

Respondents were asked to give the number of staff in their organisation employed in Data Processing Management, in Systems, and in Programming. They were then asked to project these figures two and four years forward. A summary of these figures is given in Table 4, except that no figures are reported here for 1982. Most respondents considered themselves unable to make meaningful predictions that far into the future. The expected overall increase for 1979 and 1980 is approximately 10% per annum. Of the components, the greatest expected increase (17% per annum) is in the systems field. Programming numbers are expected to increase by 7% per annum, and managers by 6.5%.

TABLE 4: EDP STAFFING 1978, 1980

	1978	1980
DP Management	57	65
Systems	88	120
Programming	150	172

2.2.2 Proportion of Qualified Staff

Respondents were asked to indicate the various levels of education and training received by their staff. Results are shown in Table 5. As in Table 3, actual figures are given, together with percentages of this sample, and percentages obtained in the larger sample reported in (1).

TABLE 5: NUMBERS OF EMPLOYEES IN ORGANISATIONS SURVEYED

	Numbers in this survey	Percentage of this survey	Percentages in larger population (1)
Numbers of staff having degrees	75	25	27
Numbers of staff with partially completed degrees	28	10	19
NZ Certificate in Data Processing	15	5	4

2.2.3 Selection Criteria for Employment

Respondents were asked to rank, from a limited selection, the criteria on which they placed most emphasis when considering an applicant for a position as systems analyst or as a programmer. The results are given in Table 6. Where individual criteria are given the same ranking (i.e. Degree in Computing, Degree in Business Studies and Polytechnic Diploma are all ranked "3" in the programmer list) the sample data showed no significant difference between those criteria.

TABLE 6: SELECTION CRITERIA FOR DP STAFF EMPLOYMENT

SYSTEMS		PROGRAMMING	
Rank		(1 highest)	
1.	Relevant experience in own organisation	1.	Relevant experience in own organisation
2.	Relevant experience in another organisation	2.	Relevant experience in another organisation
3.	Communication skills	3.	Degree in computing
4.	Management potential	3.	Degree in Business Studies
5.	Degree in Computing	3.	Polytechnic diploma
5.	Degree in Business Studies	6.	Manufacturer's course
5.	Polytechnic diploma	6.	Communication skills
5.	Manufacturers course	8.	Management potential

Clearly, experience is the most attractive attribute that an applicant can possess. For systems staff, communications and management skills are of substantially greater importance than formal academic qualifications, though for programmers the opposite is true. It should be noted that, while the desire to recruit from within one's own organisation is understandable, in terms of loyalty to the firm, etc., in practice this is rarely possible in most firms. It should also be noted, that the contrast in employment selection criteria between analyst/programmers becomes strange when it is realised that the main group of entrants to the systems sphere, are programmers (1). Few of these criteria are obviously developed by the programming task.

2.2.4 Topics in which DP staff should be trained

The relative importance of material to be taught people in data processing was assessed in two ways. Firstly, managers were asked to list the five most important topics in which new recruits to their department should be trained. The results are shown in Table 7.

Respondents were then asked to consider a list of 37 topics and to grade their attitude to applicants having skills or training in that area. Grades assigned were on a five point scale (from 0 - of no value or interest, to 4 - of high value, demonstrated proficiency required). Grades were assigned separately for selections of staff for programming and systems positions.

TABLE 7: THE MOST POPULAR TOPICS IN WHICH ENTRANTS TO DP ARE TRAINED

RANK (1 highest)	TOPIC
1	Business Systems
2	Cobol
3	Human Communication
4	Computer and Peripherals (as appropriate to the specific installation)
5	Data Base Techniques
6	Program Structure and Design

In Table 8 the topics are listed in order of importance. It should be noted that, due to the relatively small number of respondents (20), the positioning of topics should not be regarded as absolute. Not too much notice should be taken of one topic being placed two or three positions higher than another.

One unfortunate omission in the topic list was RPGII. It is unlikely in this sample that it would have scored very highly. The low scoring of PL/I was to an extent explained again by the sample selection. All the installations visited were Cobol programming shops.

Overall, the respondents emphasised the importance of the environment in which data processing operates (e.g. File Design, Batch Processing techniques, Telecommunications, Data Base Techniques). It is clear, and to be expected, the more esoteric and theoretical aspects of computer science are seen as of little or no immediate interest by data processing managers. For programmers, Cobol was considered the only language relevant in the organisations that were sampled, with the four other high level languages being relegated to the last seven positions in the programmer list. Algol (that favourite of many computer scientists) was placed last in both lists.

Discussion on what computer language to teach to undergraduate students often becomes quite heated in academic circles. Cobol is frequently criticised but Brown's comment (2) is interesting:

> "A typical classified section of the Los Angeles Times contains perhaps 50 ads for programmer jobs that mention Cobol, 5 that mention FORTRAN, and 1 that mentions PL/I, with seldom any mention of ALGOL, BASIC and APL".

Our respondents confirmed this view. The high score of

TABLE 8: A GRADED LIST OF TOPICS INCLUDED IN QUALIFICATIONS FOR PROGRAMMER APPLICANTS / SYSTEMS APPLICANTS

(Gaps in the list indicate meaningful breaks in the scores assigned)

PROGRAMMING

Cobol
Program Structure and Design

Human Communication
Batch Processing Techniques
Data Base Techniques
File Design
Computers and Peripherals
Applications Software
Telecommunications

Systems Analysis and Design
Data Preparation Equipment
Data Preparation Techniques
Operating Systems
Business Systems
Projects
Job Control Languages

Systems Programming
Management Information Systems
Financial Accounting
Programming Languages
Mini-computers
Social Implications of Computing
Assembler Programming
Mathematics
Statistics

Business Psychology
Economics
Information System Theory

Compiler Construction
Microprocessors
PL/I
Fortran
Basic
Algebra
Artificial Intelligence
Calculus
Algol

SYSTEMS

Systems Analysis, Design and Implementation
File Design
Batch Processing Techniques
Human Communication
Application Software
Data Preparation Techniques
Business Systems
Data Base Techniques
Computers and Peripherals
Management Information Systems
Telecommunications
Data Preparation Equipment and techniques
Projects
Cobol

Financial Accounting
Program Structure and Design

Minicomputers
Statistics
Operating Systems

Business Psychology
Social Implications of Computing
Job Control Languages
Information System Theory
Economics
Mathematics
Programming Languages

Micro-processors
Systems Programming
Algebra
Basic
Assembler
Calculus
Fortran
Compiler Construction
Artificial Intelligence
PL/I
Algol

Human Communication is significant also. It's clearly important for systems personnel to communicate well in both a written and oral manner. It is also thought to be important that programmers have similar skills. This is particularly true when it is known that at least two thirds of entrants to the systems field are from application programming (1).

2.2.5 General Application

Tables 3 and 4 contain, for comparison, proportions in categories obtained in this survey and in a larger one reported elsewhere (1). The latter is claimed to be representative of the New Zealand data processing profession. In Table 3 (relative proportions of staff in various functions) and Table 4 (relative proportions of those with specific qualifications) the close agreement between the surveys suggests that the 20 data processing establishments surveyed are a reasonably representative microcosm of the New Zealand computing milleau. On that basis attitudes reported by their management can be taken as representative of the majority of DP management in New Zealand.

2.3 Comparison with a Similar Survey Conducted in Australia in 1976

A survey of "Industry requirements for Data Processing courses" was conducted by Whittet (3) in Australia in 1976. The topics that were included in this survey were essentially those used by Whittet although, for use in the New Zealand environment, more topics were added. The survey reported here also considered relevance in both the systems and the programming areas ranking "topic preferences" of Australian DP management as obtained by Whittet is included as Table 9 of this paper. The request to the respondents of Whittet's survey was:

> "Rank 31 subjects for inclusion in a course in Data Processing"

as compared with the reque[illegible] given by the present authors:

> "In the attached list of topics, please grade your attitudes to the inclusion of such material in employee's qualification using the 5-point scale 0 (low) through 4 (high)."

Of the first 10 topics of Table 8 (of importance to applicants for systems work), 9 of these were included as topics in Whittet's paper, and all 9 were ranked in the first 12 of Whittet's list. In general, the results of both Whittet's and this survey emphasize the importance of DP environmental factors, together with specific skills that are immediately useful.

There is a surprisingly high correlation between both Whittet's and the author's rankings considering the difference between the New Zealand and Australian data processing industry, the differences in the tertiary educational systems, and the two year gap between the surveys.

There is little doubt about which topics are regarded by Data Processing management as important, moderately valuable, and irrelevant!

TABLE 9: TOPICS RANKED BY AUSTRALIAN DATA PROCESSING MANAGERS

Ranking (1 highest)	Topic
1	Systems Analysis, Design and Implementation
2	Computers and Peripherals
3	COBOL
4	File Design
5	Business Systems
6	Management Information Systems
8	Data Preparation Equipment and Techniques
9	FORTRAN
10	Human Communication
11	Applications Software
12	Batch Processing Techniques
13	Job Control Language
14	ASSEMBLER
15	Data Preparation Techniques
16	Systems Programming
17	Financial Accounting
18	Mini-Computers
19	PL/I
20	Projects
21	Basic
22	Mathematics
23	Statistics
24	Operations Control
25	Micro-Processors
26	Business Psychology
27	Algebra
28	Social Implications of Computing
29	Calculus
30	ALGOL
31	Economics

3. MOTIVATIONS, DREAMS, AMBITIONS AND REALITY

3.1 The Overall Pattern

The shortage of trained, experienced data processing staff has been a characteristic of both the Australian and the New Zealand employment market for a number of years. In spite of this shortage, coupled with a tightness of budgets, and a desire to use the installed data processing facility as efficiently as practicable, employers are still wary of educated but inexperienced staff. Recruitment and training of such staff involves both a degree of risk and a long-term planning capability that seems unacceptable to the data processing environments. Outside government departments, training budgets for data processing staff appeared to be slim. It is therefore not surprising that data processing managers want experienced personnel. Pressures exerted on data processing management are essentially pragmatic. High quality systems are to be delivered at known cost, in acceptable time frames. Meeting such objectives requires experienced personnel often working to substantial pressure in small groups. There is likely to be little time to devote to training junior staff.

3.2 The Situation as seen by the Full-time Student of Computing

Many full-time students of computing were introduced to the subject whilst still at school through the Applied Mathematics Seventh Form subject or their local computer club. They find computing imaginative, exciting and stimulating. They enjoy the fairly loosely-controlled problem solving environment. They believe that a career in computing would be both intellectually demanding and financially rewarding.

Rather than enter full-time employment after Seventh Form (virtually all such students have "University Entrance") they typically choose to progress their education either at a Polytechnic or at a University. There is still a great deal of social pressure that favours academic qualifications - whatever the discipline.

After completion of their tertiary education (an education which may well have increased their enthusiasm for computing) such students find that the computing market place is typically for applications programmers, or for experienced systems analysts. If they do gain employment into the applications programming field, it will typically be as a "Trainee". There, the reality of the job pattern rewards attention to detail and conformity to standards, rather than mental agility. The survey reported here shows that, for example, a degree in Computer Science is only marginally preferable to a degree in any other discipline.

3.3 The Situation as seen by Teachers of Computing

Both Polytechnics and Universities teach computing. However their enrolment procedures, their student motivation and staff attitudes tend to be quite different. In Polytechnics, students generally enrol in a Diploma course, which has a high vocational element. In a University, students enrol in a degree program in which there is typically a core of mandatory papers, supplemented by a wide range of electives, and opportunities to major. For teachers of data processing within Polytechnics, the dream is for an industrial environment which accepts their "products" as both trained and partially experienced. The reality apparently is of an attitude which regards Polytechnic diplomates as of only marginal value in staff selection.

For teachers of computer science in universities, the dream is of a different type.

The ideal situation is one of limitless hardware for research and experimentation, and of substantial numbers of enthusiastic students, hopefully each with a strong "research" motivation. Through a desire to contribute to the growing body of computing knowledge in the university and a general user community, university staff tend to dream of "unsullied virgin areas" of the discipline, where their footprints will readily be recognised, remembered and rewarded! Of course, the reality is different! Large numbers of students, often with little interest or enthusiasm for the subject, attend computing classes. Despite views to the contrary, hardware acquisition for teaching or research is still difficult, with strong cases being needed. However, as section 1 of this paper suggested, when university staff select topics to teach in depth at advanced levels, their academic interests cause them to tend to select the more technical and esoteric areas (i.e. compiler construction, operating system design, artificial intelligence) rather than some of the problems that are immediately applicable in the reality of day-to-day processing. Essentially, many university teaching staff seem to dream of not teaching pragmatics, but of working in subject areas selected only by their own particular interests.

3.4 The Situation as seen by the Data Processing Managers

The survey reported in section 2 of this paper described a situation o pressures and therefore of pragmatics. To employers, an applicant for a data processing position represents a solution to a short-term job need rather than a long-term career prospect. In a recent issue of Pacific Computer Weekly, under the headline "Staff Shortage - but no jobs for trainees", Davey (4) was reported as saying:-

> "Despite the massive shortage (in Australia) of computer staff, many trained programmers and operators cannot get jobs .... The magic figure seems to be two years experience to get a job. But where do trainees get this experience, if no-one will take them on?"

The dream, then, is that junior programming and systems vacancies can be filled by "two years experienced" staff. The reality in some centres in New Zealand, seems to be that this experience and training is gained at the expense of large employers (the government departments and the machine companies) and that the staff move on after acquiring the mandatory 2 years experience.

3.5 The Conflicts of "Education" and "Training"

It seems unreasonable to expect the educational institutes to provide "experience". The reality of the nature of most organisation's data processing operation, and of the many aspects of experience that can usually only be simulated in a teaching environment, virtually ensures that products of the New Zealand full-time tertiary institutions will be qualified, but only partially trained. Inevitably, they will be almost totally inexperienced in many areas of data processing.

The provision of an immediately saleable meal ticket is rarely seen as the function of a university. Topics for inclusion in courses are typically selected on the basis of their overall contribution to the education of the undergraduate, as much as for the relevance of such material to the world of computing.

After graduation, it is seen as the task of the employer to build on this overall education and to train the student in the specific skills that the organisation may require.

As we have noted in section 2, the requirements regarded as important for new entrants to data processing by different data processing managers are essentally pragmatic, and skill oriented. For most staff who design and teach courses in computing in universities, there are a wider range of objectives of which three are perhaps the most pressing, viz:

> To give to as many undergraduates as practicable, an all-round appreciation of computing
>
> To allow specialist students in computing to develop problem-solving skills in a variety of situations, in order to support their education
>
> To give to certain other specialist students (e.g. in Accountancy or Technology, or Management) an in-depth awareness of the impact of computing on their discipline.

In a real sense, the satisfaction of this complex of objectives by a very limited number of papers, means that substantial compromises are required. This further removes the end-product from the ideal of the employer.

A recent editorial in Computerworld (5) speaks to this

> "Most computer science programs do not intend to produce DP experts. While many computer science students do graduate with DP skills and most computer science graduates - at the bachelor degree level - do seek DP jobs, DP is not computer science any more than carpentry is architecture".

It must be remembered that many, in fact the majority, of those involved in data processing are non-graduates. They are also doing an outstanding job. Again quoting the same editorial (5):

> "Many graduates of university computer science programs are willing and able to perform well in applied DP positions, but employers are mistaken if they assume only such graduates are qualified to take these jobs.
>
> Thousands of experienced programmers in this country demonstrate every day it is possible to acquire applied DP skills without attending universities at all. However, there is a crying need for more applied DP training at professional institutes and trade schools or within such disciplines as business administration and applied engineering."

It is interesting to note, however, that a demonstrable knowledge of computing or data processing is seen as a very attractive element in a degree or diploma in a non-computing discipline. As one example (there are many others) the Bachelor of Business Studies offered at Massey University offers a general education in Business, together with an opportunity to specialise in a number of areas. Of the graduates, those with a dual major, in both Finance and Accounting and Data Processing, seem to be amongst the most sought after. Perhaps part of the reason for this is the existence of a professional body, the New Zealand Society of Accountants, which co-operates with the university in planning courses and accepts the degree as part fulfillment of its formal membership requirement. Graduates with the Massey Business degree are educated and partially qualified, and therefore, apparently, attractive. The data processing courses are viewed as a useful bonus, which substantially increases the "field of vision" of the accountant or manager.

4. PROBLEMS OF COMMUNICATION

Such an analysis of the different aims and attitudes of the three elements in computing education and training (student, teacher and employer) suggests a breakdown in communication. Certainly the teaching institutions are frequently criticised for being out-of-touch with the reality of data processing.

In his keynote address to the Fifth New Zealand Computer Conference in 1976, Mr Ashley Goldsworthy (6) had this to say of his attitude (in Australia) to computer science graduates:

> "....I don't want Computer Scientists, I don't need people who know lots about operating systems or programming languages or those sort of things. I need MBA's (Masters of Business Administration, usually a three year part-time post graduate course in Australia) who understand business, who understand finance, who understand economics, and who understand enough of computing to be able to design systems and implement systems using the computer as a tool .... I think it's a tragedy when people do undergraduate studies, 80% Computer Science - they're almost non-employable as compared with the student who has done substantial studies in computer science, data processing, information processing or whatever you want to call it...."

Mr Goldsworthy failed to recognise both that "majors" in computer science in most New Zealand universities will in general do less than 30% of their papers in computing, and also that the universities are still primarily involved in education, not in the provision of pre-digested experience.

The relatively low correlation between expectations of students of computer science, the short term requirements of employers of such students, and the educational objectives of courses in computing as seen by professional educators is a matter of real concern. The evidence of shortages of trained computing staff and criticism of computer course material has been discussed in this paper. What solutions can be suggested?

Clearly, some increased co-operation between the data processing industry and the teaching institutions could have immediate benefits to students. Visits to computer installations or seminars by data processing practitioners, are easy to arrange, and are usually well appreciated. The extreme of this co-operation, as practiced in UK, where many polytechnics and technical institutions place their full-time students into industry for two periods of six months each, in a three or four year course, this extreme is difficult to achieve, and may well be inappropriate to New Zealand. This is done by some technical institutions in New Zealand in the form of day/block release schemes. The results of such schemes are:

Awareness by the students of the reality of life in a data processing shop

Awareness by the teaching staff of the needs of the data processing management

Awareness by the employer of the capabilities and limitations of the students

With a substantially increases awareness of the needs and pressures of the other elements in the mix, the effectiveness of communication between teachers, students and employers can be markedly improved.

Certainly, there is a tendency at the moment by each section not to recognise the requirements and objectives of the others. Students often fail to realise their short-comings. Employers fail to recognise the many objectives of tertiary education. Educators sometimes avoid the reality of data processing, as though it were not quite respectable. Some of these emphases could clearly be corrected. And the New Zealand Computer Society could help. Students could be encouraged to attend branch meetings, perhaps through the preferential class of membership currently used by certain branches. Employers, possibly through the Computer Society Branch Committees, could provide a formal organisation with which teaching staff in computing could liaise with a view to selecting (where educationally appropriate) material for inclusion in courses. If computing qualifications were made to be more relevant, and of course were seen to be relevant, and therefore were to become mandatory (or at least useful) for job applicants, then more employers may become aware of the very real talent and enthusiasm of many of the computer science and data processing students who are currently enrolled in courses.

A comparable shift in emphasis, from the techniques of programming, to the rather more subtle and complex area of systems work, together with a widening of the market for such courses into the areas of business or commerce undergraduates, would go some way to correcting the ignorance of the data processing potential of computer systems that is still a regrettable feature of many graduates.

Computers exist to provide answers to peoples' problems. Graduates from courses with a strong computing (or data processing - the term should be interchangeable) emphasis should not graduate with the dangerous combination of a strong knowledge of computer equipment and software, and an ignorance of (or even apathy towards) the situations which justify computer acquisition.

There is nothing incompatible with the use of computing technology as a framework for education, and the awareness that computers exist not as

4

ends in themselves, but as a means of supporting people in organisations to operate effectively.

ACKNOWLEDGEMENTS

The authors would like to thank the following:

The New Zealand Computer Society, Massey University and Victoria University of Wellington for their support of the work reported here.

Those who enabled the survey of computer education by filling in the course description forms.

The data processing managers who bared their souls by telling us what they expected of potential recruits (and all of whom asked "what is artificial intelligence?")

REFERENCES

1. Boswell, C.R. and Melhuish, P.J.
Data Processing Personnel and their careers - a profile

Proceedings of the Sixth Computer Conference of the
New Zealand Computer Society (1978)

2. Brown, G.D.
Advanced ANS COBOL with Structured Programming

Wiley Interscience, (1977)

3. Whittet, L.T.M.
Data Processing Courses - a survey of Industry Requirements
Proceedings of the Australian Colleges of Advanced Education
Computing Conference, 1976.

4. Davey, D.
reported in Pacific Computer Weekly, April 28, 1978

5. Editorial
A Sad State of Confusion
Computerworld, Vol XII, No.21, P.24 (1978)

6. Goldsworthy, A.W.
Proceedings of the Fifth Computer Conference of the
New Zealand Computer Society, Volume 2, pp. 67-89 (1976)

APPENDIX 1

QUESTIONNAIRE SENT TO ESTABLISHMENTS TEACHING COMPUTING

NAME OF INSTITUTION/DEPARTMENT ....

....

NAME OF RESPONDENT ....

BRIEF DESCRIPTION OF COURSE CONTENTS:-
(Attach syllabus if available) ....

....

....

....

....

AVERAGE NUMBER OF STUDENTS ENROLLED ....

DURATION OF COURSE ....

NUMBER OF TIMES PER YEAR THAT THE COURSE IS OFFERED ....

TUITION FEE
(if appropriate) ....

NORMAL STUDENT ENTRY REQUIREMENT ....

METHOD OF TUITION
(please delete where method not used) LECTURE TUTORIAL SEMINAR

CORRESPONDENCE PRACTICAL "LABORATORY" SESSION

TO WHAT QUALIFICATION (if any) DOES SUCCESS IN THIS COURSE CONTRIBUTE? ....

....

....

APPENDIX 2.

SAMPLE PAGES FROM "COMPUTER EDUCATION IN NEW ZEALAND"

Institution : University of Auckland
Department : Board of Computer Studies
Title : Compiler construction.

The Lectures will cover the design and implementation principles of compilers and interpreters, so that sutdents may build up the background needed to construct a compiler. The couse is intended to be practical rather than theoretical: each student will be required to undertake a project, e.g. the project will involve implementing a compiler for a given language. At least six languages will be specified - each student will select a language, and (individually) write, test, and document a compiler (or interpreter) for for it.

Duration : 23 weeks
Fee : $20.00
Prerequisites: Stage 2 "Programming Languages" plus "Information Structures".
Teaching Methods: Lectures, Tutorials, Practical Work.
Contributes to: B.A., B.Sc., B.Com, etc.

Institution : Auckland University
Department : Board of Computer studies
Title : Computer operating systems.

This couse should give students an understanding of the underlying principles of operating systems, as well as some experience in developing large computer programs.

Duration : 23 weeks
Fee : $20.00
Prerequisites : Stage 2 "Programming Languages" plus "Information Structures".
Teaching Methods: Lectures, Tutorials,
Contributes to : B.A., B.Sc., B.Com., etc.

APPENDIX 2 (cont)

Institution : University of Auckland
Department : Department of Management Studies
Title : Management Information Systems.

Stage III degree paper, approximately 1/20th degree, computer technology applied to the managerial decision - making processes of planning, organising and controlling.

Institution : Christchurch Technical Institute
Department : Department of Applied Science
Title : Computer Methods.

TCA (Year 5) NZC Stats and NZC D.P. Main emphasis on FORTRAN TCA course to give a deeper coverage of FORTRAN

Duration : 1 year
Fee : $15.50
Prerequisites : An NZC course or reasonable maths background.
Teaching Methods : Lectures, Tutorials, Practical work.
Contributes to : NZC Stats, NZCDP, TCA Certificate of Attainment or CTI C of A.

Institution : Christchurch Technical Institute
Department : Department of Applied Sciences
Title : Programming with BASIC

2 Terms of 2 hours per week. Students own programs on a PDP11 up to level allowed by Multi User Basic. Course is on the use of the language rather than Scientific Techniques.

Duration: 1 or 2 terms
Fee: $13.00
Teaching Methods: Tutorials, Practical work
Contributes to: A CTI Certificate.

APPENDIX 2 (cont)

Institution : Christchurch Technical Institute
Department : Department of Applied Science
Title : Introduction to Data Processing

Elementary levels only of computer hardware uses of computers in industry and business, systems investigation, etc. No programming.

Duration: 38 weeks
Fee: $12.00
Teaching Methods: Lectures, Tutorials.

APPENDIX 3 — QUESTIONNAIRE SENT TO DATA PROCESSING MANAGERS

the New Zealand
Computer Society

Careers in Data Processing — 1977

Massey
University

(A Survey)

DATA PROCESSING COURSE EVALUATION
A SURVEY

Note — when employees are referred to, the term is to be understood to apply
to D. P. Management
to Systems Analysts
and Programmers —
both Trainee and through to Senior staff.

Organisation Title ....

Name of Respondent .... Date ....

Title of Respondent ....

1. How many employees of the following categories do you employ, or intend to employ?

	Now	1980	1982
D. P. Management			
Systems			
Programming			

2. For each of the three categories currently employed — estimate the **proportion** who are trained or qualified in the following categories.

	D.P. Management	Systems	Programming
University degree —			
Completed — Computer Science			
Part completed — Computer Sc.			
Completed — Other discipline			
Part completed — Other Disc			
Polytechnic diploma/certificate			
N.Z. Certificate in Computing			
Computer manufacturers courses			
Internal training (own organisation)			
Training from previous employment			

C. R. BOSWELL
Director
Computing Services Centre
Victoria University of Wellington

P. J. MELHUISH
Senior Lecturer
Computer Science Dept.
Massey University
Palmerston North

APPENDIX 3 (Cont'd . . .)

3. Selection criteria for employment of personnel.

Prospective employees bring a variety of experience, qualifications and aptitudes. Some of these are listed below. Quite a few are fairly general and vague. Please rank these, for the categories of Systems Analyst and Programmer, as though you were interviewing for a position in your organisation.

	Systems	Programming
Degree in (Computer Science major)		
Degree in (Business Studies major)		
Polytechnic diploma in D.P.		
Manufacturers computer course		
Relevant experience in own organisation		
Relevant experience in another organisation		
Management potential		
Communications skills		

4. What in your view are the five most important topics, in which staff beginning a career in your organisation, in data processing should be trained?

1.
2.
3.
4.
5.

5. Assuming that you have decided to take note of a degree or diploma that includes some computing or computer science papers, the matter of subject matter, in papers taken, becomes important. In the attached list of topics, please grade your attitudes to the inclusion of such material in employee's qualifications, using the following 5 point scale:-

0	of no value or interest
1	of only moderate and occasional value
2	useful material
3	of real value, some demonstrated proficiency preferred
4	of high value, demonstrated proficiency required

APPENDIX 3 (Cont'd....)

4.

Ref	Topic List	Applicants for Programming GRADING - RANGE	Systems 0 THRU 4
1.	Mini-computers		
2.	Micro-processors		
3.	Computers and Peripherals		
4.	Data Preparation Equipment and Techniques		
5.	COBOL		
6.	PL/I		
7.	FORTRAN		
8.	BASIC		
9.	ALGOL		
10.	ASSEMBLER		
11.	Job Control Languages		
12.	Systems Programming		
13.	Applications Software		
14.	File Design		
15.	Data Base Techniques		
16.	Batch Processing Techniques		
17.	Data Preparation Techniques		
18.	Systems Analysis, Design and Implementation		
20.	Mathematics		
21.	Algebra		
22.	Calculus		
23.	Statistics		
24.	Financial Accounting		
25.	Economics		
26.	Business Systems		
27.	Human Communication		

APPENDIX 3 (Cont'd....)

Ref	Topic List	Applicants for	
		Programming	Systems
		GRADING - RANGE 0 THRU 4	
28.	Management Information Systems		
29.	Social Implications of Computing		
30.	Business Psychology		
31.	Projects		
32.	Program Structure and Design		
33.	Telecommunications		
34.	Operating Systems		
35.	Compiler Construction		
36.	Artificial Intelligence		
37.	Information System Theory		
38.	Programming Languages		

VICNET:
AN X.25 PACKET SWITCHING NETWORK

AUTHOR: S.W. BOWLES
Department of Information Science
Victoria University of Wellington.

ABSTRACT:

Three organisations housed at Victoria University of Wellington have undertaken to interlink their own computers using the relatively new software technology known as packet switching. This has been accomplished using the international standard user interface called X.25. The result is a practical service for users of present and future VUW computers, and a testbed for developing expertise inside New Zealand in an area which may soon prove of interest to public communications.

KEY WORDS AND PHRASES

Packet switching, node, host, X.25, network, Network definition language, communication protocols, mini and micro-computers.

C.R. CATEGORIES

2.12, 2.43, 2.44, 3.81, 3.83, 4.35, 4.49, 6.29

VICNET:
AN X.25 PACKET SWITCHING NETWORK

S.W. Bowles

Department of Information Science
Victoria University of Wellington

ABSTRACT

Three organisations housed at Victoria University of Wellington have undertaken to interlink their own computers using the relatively new software technology known as packet switching. This has been accomplished using the international standard user interface called X.25. The result is a practical service for users of present and future VUW computers, and a testbed for developing expertise inside New Zealand in an area which may soon prove of interest to public communications.

1. BACKGROUND

Definition

What is a packet switching network? It is a mesh of interconnected mini- (or micro-) computers, connected to each other, and at the edges of the mesh, to user machines or Hosts. Each mini is called a "node" of the packet switching network, and is dedicated to data communications processing.

When one of the Hosts wishes to converse with another Host, it does so by sending "packets" or small blocks of data to its local node - the one it is directly wired to - along with some additional information sufficing to address the packet to the other Host. It is then up to the "mesh" - the packet switching network - to send the packets to that other Host, by whatever devious route or routes seem optimal at the time. See Figure 1.

Motivation

Why is a packet switching network a good idea? Well, lines are expensive, not very reliable, and generally are under utilised. If we had had to lease a physical circuit from Host P to Host Q, we would have paid for it, effectively, twenty-four hours a day, sixty minutes on the hour, sixty seconds to the minute; it is doubtful we would actually use all that raw data communications capability. Besides, that single line might just break down some day. A leased line is an all or nothing proposition: if you don't use it all, you feel you are wasting your money, and if it goes down, you have got nothing at all!

With a packet switching network we still use standard leased lines between each pair of connected nodes (the nodes needn't all be connected to all others though). But these lines, and their cost, are shared - efficiently and safely if the network software is right - by the many subscribing users of the net, and if one or two of them become broken or faulty at some point, that same software is smart enough to reroute our packets around the problem. Indeed, if all the lines are working, cleverly constructed software in the nodes can push our effective throughput higher than the speed of any one physical leased line, by routing a series of packets in parallel along different available lines!

A final benefit is simple communications flexibility. Who would accept the idea that, when ordering a phone, he must enumerate his intended correspondents? But that is where we are with leased lines! With a packet switching network, once you join the packet switching community, you can "dial up" any other subscriber. More of this and its effect on data processing standards later.

So packet switching is potentially cheaper, more reliable, faster, and certainly more flexible than other currently available data communcation technologies (we are now starting to use that word about software techniques). Why didn't we think of this before?

HISTORY: 1966 - 1978

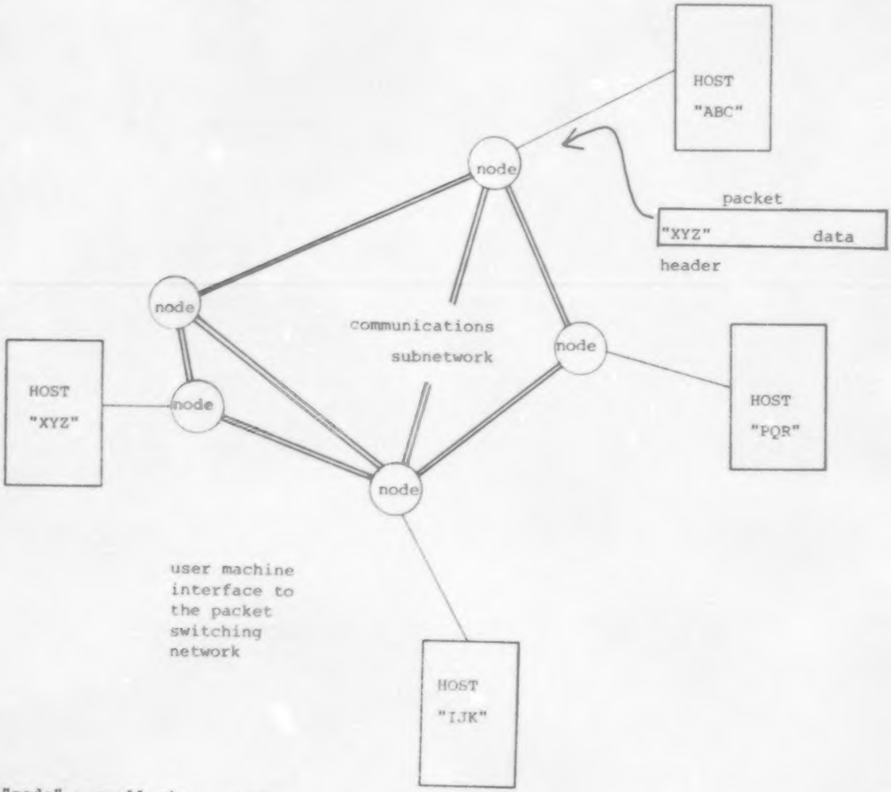
One of the reasons is that minicomputers used to be expensive, and people thought it a bit extravagant if anyone suggested that several of them be dedicated just to data communications. When Paul Baran of the RAND Corporation in California first suggested it in 1960, it was as a means of fail-safe military communication, where cost was not much of an object; and even in 1966, when L.G. Roberts proposed that the Advanced Research Projects Agency (ARPA) of the United States Department of Defense undertake a project to interlink several large Hosts coast to coast via packet switching, it was only because times were good, especially for defense projects, that it got off the ground. But by now ARAPANET is almost a dinosaur; when in 1978 we look at minicomputer costs versus line costs and consider how well developed the software methods are - well, everyone should have a packet switching network in his backyard.

And such is nearly the case. Following close on ARAPANET, we have Cyclades in France and EPSS in the U.K., and GMDNET in West Germany. These networks were experimental and theoretical; but the entrepreneurs and the public communications men weren't far behind: Transpac in France, Telenet in the U.S.A., and public networks in Germany and the U.K. In Europe, the expertise is being pooled in Euronet, the European Informatics Network, by the Common Market. Everyone is working on the same thing - and rest assured, they aren't agreeing very much on the best ways of doing it!

Which brings us to another reason why it wasn't all done in the sixties. It's very difficult to set two computers to speak the same language, especially, and notoriously, if they were made by different companies. Even to set them to speak the same data communications line discipline can be prohibitively hard! What is needed are standards, and in

October of 1976 the International Consultative Committee for Telgraphy and Telephony, or CCITT, passed Recommendation X.25, entitled

> "...the interface between data terminal equipment and data circuit terminating equipment for terminals operating in the packet mode on public data networks",


HOST
"ABC"
node
packet
"XYZ"
data
header
node
communications
subnetwork
node
HOST
"PQR"
HOST
"XYZ"
node
node
user machine
interface to
the packet
switching
network
HOST
"IJK"

"node" = small micro or mini
"HOST" = user computer
——— = conventional data circuit
═══ = wideband leased line

FIGURE 1 - Definition of a Packet Switching Network

which specification [1], on about the thirteenth reading, tells you how a user or "subscriber" machine is supposed to relate to the node of the network he is connected to - how it "addresses" packets to another Host, how it recovers from line errors, and all.

This may not sound like much - it's not - but in terms of the movement for standards in the computer industry it may prove to be the shot heard around the world.

For this is only the beginning. If we want to effectively connect many large Hosts together, it is positively frightening to think how many differences will have to be smoothed over, in file accessing techniques, job control languages, application and utility parameters, operating system idiosyncracies to name but a few. Still, the potential benefits are enormous, and this means there will be enormous pressure for that "smoothing" to be done. Packet switching may just be the battleground for East to meet West in the computer world - and the sparks will fly!

2. OBJECTIVES AND CONSTRAINTS

It is natural, then, that when three Victoria University based organisations met in September of 1976 to discuss data communication developments at Victoria, the conversation should turn to packet switching and to X.25.

The existing Victoria network was of the usual "star" type. The Computing Services Centre (CSC) of the University owns a Burroughs B6700, designed to support batch and interactive processing; a separate Data Communications Processor handles all traffic to other machines and terminals, and may be programmed using a "Network definition language" or NDL. The Applied Mathematics Division (AMD) of the D.S.I.R., housed on the Victoria campus, owns a Hewlett-Packard 2100, supporting several peripheral devices and thirteen local and remote terminals, any of which may either use the 2100 Basic interpreter or hook through to the B6700 via a half duplex 9600 baud line. The Information Science Department (INFO) owns a Digital Equipment Corporation PDP11/10 with thirteen serial ports, each capable of servicing either a terminal or another machine; one port takes a 9600 baud half duplex line to the B6700. See Figure 2-1.

A fourth machine entered the picture in 1977 when Information Science acquired a Motorola M6800 system from a local firm, Associated Processor Systems Limited. This mini-size microcomputer system,very cheap and eminently well suited to small scale data communications, was quickly accepted as a valuable addition to the network - whether as a small Host, or within the net as a communications node. In fact the CSC also acquired in 1977 a small M6800 for terminal concentration applications; additional M6800's have been ordered in 1978, by Info and CSC, to serve as network nodes.

The situation in respect of the AMD HP2100 - CSC B6700 connection provided the impetus for the discussion as the then current link implementation (9600 baud half duplex polled) was inflicting unacceptable inefficiences on both systems and was insufficiently flexible to meet projected needs.

The Information Science Department was called in for consultation, as it had some experience in these matters from its recent work with Massey University on the fledgeling inter-University packet switching network called Kiwinet [2]. X.25 was explained and discussed, and the decision taken, in October 1976, to commence implementation of this standard, on all three machines - with a view to later development into a full fledged packet switching network, as in Figure 2-2. The project was named Vicnet.

The objective of the effort were to be:-

1) Standardization. We felt that the development, debugging, and support of data communication protocols are too difficult, and too much work, to be approached (anymore) in an ad hoc manner. By adopting a common discipline among ourselves, we stood to save programmer comprehension time, some debugging time, and much trouble-shooting time. When we reached the support phase; by jumping on the X.25 bandwagon, we stood to gain from overseas

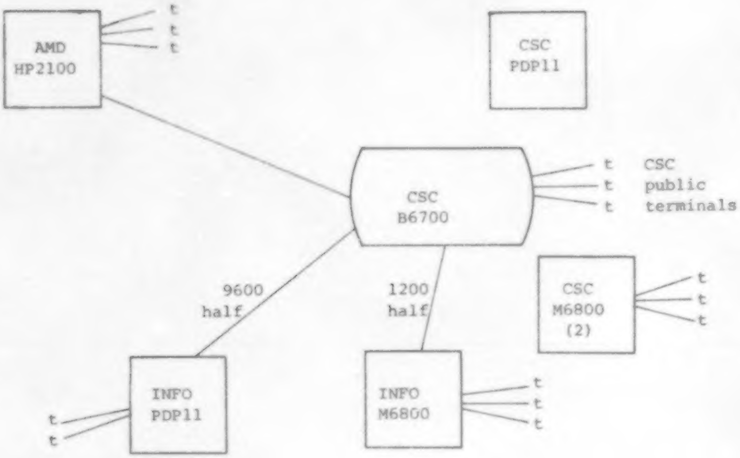

AMD
HP2100
t
t
t
CSC
PDP11
CSC
B6700
t
t
t
CSC
public
terminals
9600
half
1200
half
CSC
M6800
(2)
t
t
t
INFO
PDP11
t
t
INFO
M6800
t
t
t

FIGURE 2-1 - Original VUW "Star" Network

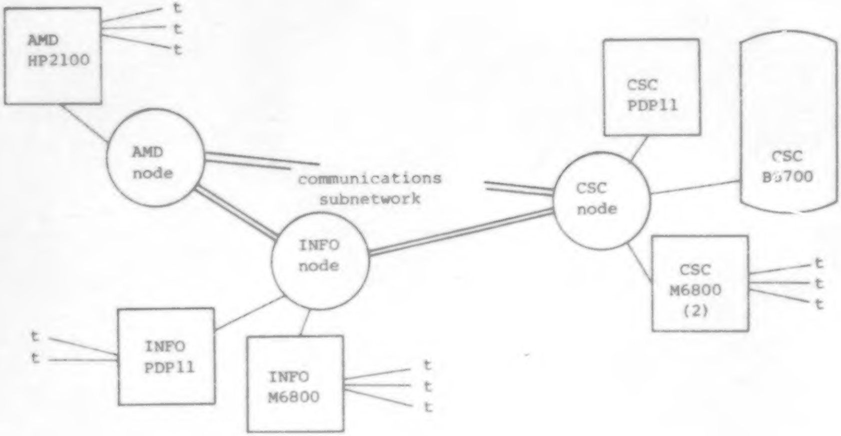

AMD
HP2100
t
t
t
AMD
node
communications
subnetwork
CSC
PDP11
CSC
B6700
CSC
node
INFO
node
CSC
M6800
(2)
t
t
t
INFO
PDP11
t
t
INFO
M6800
t
t
t

FIGURE 2-2 - Project Vicnet Packet Switching Network

research, experience, and future technological breakthroughs.

2) Speed and reliability. X.25 is compatible with full duplex operation and has been designed for efficient error recovery with a large number of independent stations.

3) Flexibility. By heading toward a common means of machine to machine communication and moreover toward a general packet switching protocol, we hoped to eventually include switching ability, so that any machine could address any other even if no direct physical connection was available.

Constraints were:-

1) Line speed and transmission mode. Quality of lines and design of existing line interfacing equipment limited us to 9600 baud asynchronous transmission.

2) Rigidity of existing software structures. Although the B6700 seems relatively well structured for data communications, the NDL (Network Definition Language) can be frustrating to use in developing machine-to-machine line disciplines. Some operating system software for the HP2100 and PDP11 needed to be rewritten from scratch, with care to avoid disrupting existing services.

3) Modest manpower resources. The work was allocated essentially to one person from each organisation, who could devote only a part of his time to Vicnet.

3. DESIGN

We began by hacking off a reasonable implementable subset of the full X.25 spec: just levels one and two, the point-to-point line discipline, suitably adapted for an asynchronous line and also for possible half duplex transmission. In this work - essentially the non-trival problem of interpreting the letter and spirit of X.25 - we were much helped by the corresponding standard from the International Standards Organisation called High Level Data Link Control. ISO's HDLC [3] was the precursor for (levels 1 and 2 of) X.25, and is, we might add, written in language more familiar to computer types.

But we also recognized the need to look ahead into packet switching generally, to carve out a future path for Vicnet and to bring our current labours into perspective. Thus it was that a survey was conducted into "Communication Protocols in Current Packet Switching Networks" [4] and a general design - a sort of five year plan - drawn up to indicate how, after this mere point-to-point discipline completed, we might smoothly move into full packet switching. We studied four major packet switching networks: ARPANET in the U.S.A., EPSS in the U.K., Cyclades in France, and GMDNET in Germany, in an attempt to arrive at a workable proposal for the design of Vicnet.

Modules

In the resulting idealised packet switching network, the switching of packets internal to the network, is done by very simple, very fast modules called Nodes, interconnected via very fast full duplex physical lines. The mesh of packet switching Nodes is called the "communications subnetwork". A Node does no "real computing" and is only hooked to other Nodes, and, possibly, to a single End-node.

An End-node is conceptually a larger, but possibly slower, module, with one line to some Node, and one or more lines to subscriber (user) Hosts. It is the interface between the Host and End-node that is pure X.25: the End-node is the network, as far as the Host is concerned, and, beginning the user/network interface, must take responsibility for charging, fault finding, and possibly for intranetwork routing of packet traffic from its Host through the network to intended destination Hosts. See Figure 3.

To accommodate those potential subscribers who have no computer, but own one or more terminals, and wish to use the packet switching network to access remote Hosts and other terminals, we supply a third module, a pseudo-Host called a Terminal Interface Process. The TIP can handle a large number of standard teletype-like terminals. The TIP and terminal look to the network just like another standard Host.

Note that we have deliberately used the term "module" and not "machine". The software techniques used in implementing these pieces allow us to run one, two, or all three of the modules in a single machine, if it is big enough; in fact, we can run several copies of each module if that is convenient, say for testing! In particular it may often be expedient to include a Node and an End-node in the same physical machine.

A fourth module, the Network Control Computer, is a single, larger machine, with a proper disk backing store, which is used by the Nodes and End-nodes to provide an automatic directory-enquiry service, to handle charging for network services, and in general to round out the network into an entity with some intelligence and long term memory. At present the NCC is little more than a concept; but it will be a necessary component in a secure, economic, powerful data communications service.

Protocols

The interface between a subscriber's Host and his local End-node is precisely X.25, and indeed this is what has been most completely implemented at Victoria so far. This protocol is discussed in the accompanying paper [10].

We have also seen no reason, then, not to adopt levels 1 and 2 of X.25 - also known as HDLC - as our protocol on any and all physical lines between machines in the packet switching network. Thus all the straight lines in Figure 3 have HDLC handling physical transmission and error recovery.

A major protocol remains: what might be called the intra- network packet protocol. That is, given that Host A has requested establishment of a "call" to Host B, what physical lines and Node machines should participate in the forwarding of the data packets, and how and when does a route get chosen; how, further, do we make sure that Host A does not saturate our network with packets of that call, if for instance B is not absorbing them at the rate A is sending them? These questions, routing and flow control, are discussed in [11].

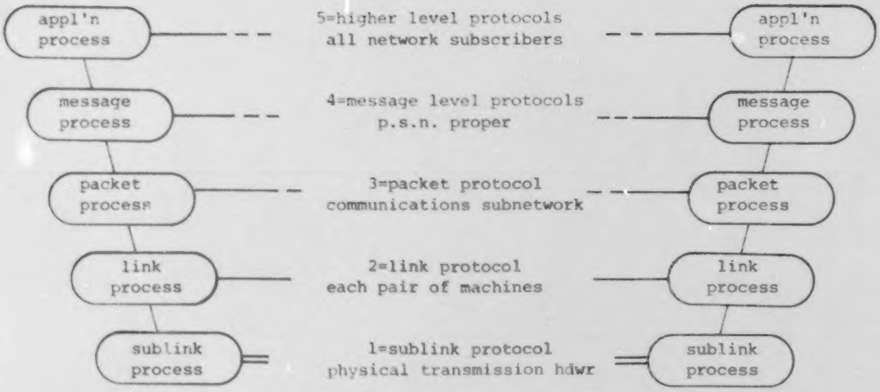

appl'n process
5=higher level protocols
all network subscribers
appl'n process
message process
4=message level protocols
p.s.n. proper
message process
packet process
3=packet protocol
communications subnetwork
packet process
link process
2=link protocol
each pair of machines
link process
sublink process
1=sublink protocol
physical transmission hdwr
sublink process

FIGURE 3-1 - Layers of Vicnet Protocol

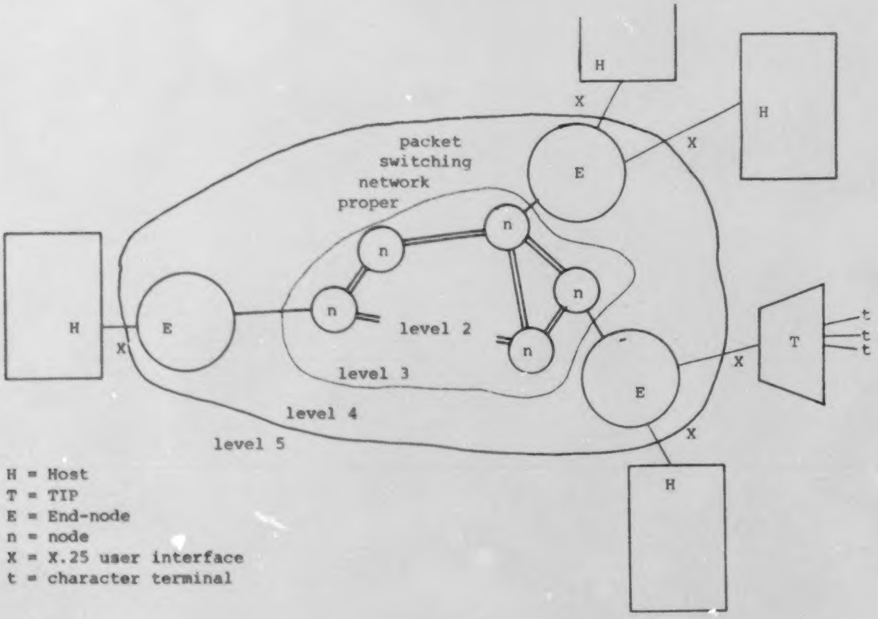

H
X
H
X
packet
switching
network
proper
E
n
n
n
H
X
E
n
level 2
n
t
T
t
t
X
level 3
E
level 4
X
level 5
H
H = Host
T = TIP
E = End-node
n = node
X = X.25 user interface
t = character terminal

FIGURE 3-2 - Vicnet Modules

Higher level protocols are needed to govern the communication between the complex operating systems and applications software in two connected Hosts, particularly in the area of interactive computing. For terminal-to-application communication, the Virtual Terminal Protocol [5] has been fully researched and designed, and partly implemented; it allows any type of terminal to converse with any other, and allows application programs to be designed and coded for interactive usage without dependence on the hardware characteristics of the terminals using them. The Vicnet Terminal was based on the EuroNet work and is, we believe, an important New Zealand "first" in itself. The other two protocols, the Interactive Terminal Protocol and the File Transfer Protocol, will be studied in 1978.

Higher still, we will need a Job Control Protocol, eventually, if human users are to make effective use of multiple and various remote machines. This subject is under separate study at Massey University [6].

4. FACILITIES

What can a subscriber do with a packet switching network like Vicnet? We present a few simple examples and then project some ideas into the immediate future.

Present Vicnet Facilities

Suppose you have bought a computer terminal with an acoustic coupler which you have planned to use to connect to your firm's computer, a hundred miles away, via a standard Post Office dialed telephone line. Then you hear of Vicnet, and you are made to understand that a Vicnet End-node, with a Terminal Interface Processor, has been installed in your town.

All you need to do is ring up the TIP, with a local telephone call, put your telephone handset down onto the acoustic coupler and wait for the TIP to say NUMBER PLEASE, whereupon you type ACME, or whatever is the Vicnet subscriber name of your firm's main office computer. After a second or two you receive CALL CONNECTED, GO AHEAD PLEASE - and proceed as you would had you dialed all the way through the Post Office telephone network.

So what have you saved? You have used the standard telephone network only as far as the local call, and this is of course negligibly cheap. How expensive is the Vicnet service in transporting lines (packets) from terminal to distant mainframe and back? Well, quantitative answers are not available yet, but the answer is "a lot cheaper". The main thing is, you are paying by the packet and a bit more for the call-request - not for the total connect times as with the telephone system.

As a byproduct of using the packet switching network, you can from this same terminal select any mainframe connected to the network, and, if you own a user code for that machine, use it just as if you were directly connected to it. You are protected from that mainframe's hardware and software idiosyncrasies - transmission speed, character representation, terminal interfacing conventions, and the like - by the Vicnet communication protocols.

Another example. Suppose you have a minicomputer in your branch office which does your local data processing; the established procedure has been to send magnetic tapes containing edited transactions into the main office, two hundred miles away, twice a week. You have been asked to investigate online methods, and you hear of Vicnet.

To use Vicnet from a machine, you do have to do some work: you have to interface your machine as a subscriber, to the nearest End-node, according to the X.25 interface standard. This involves some programming - see reference[10] - but when it is done your machine can automatically "dial up" the distant mainframe or for that matter any mainframe or terminal in the network. Any terminal on your local mini could also, with a suitable software bypass in the mini, access the net, and thus any mainframe or terminal thereto connected. And again, the data transport service promises to be much cheaper than for conventional data communications, and potentially also faster and more fail-safe.

A third and final example is in some way intermediate between the foregoing two. Suppose you are considering buying six terminals to sit in your branch office and access your central office's computer a hundred miles away. You could just use six separate dial-up telephone lines, each connecting one terminal all the way to the mainframe; or you could buy a small minicomputer and program it to "concentrate" the six down one dial-up (or perhaps leased) line to the mainframe.

But hearing of Vicnet, and the proximity of a Vicnet End-node, you realise that you can rent a TIP from the network authority, cable your terminals all to it on your premises, and lease a short line to the local End-node. Then any terminal can dial up any mainframe (or other terminal) on the net.

There are just three simple examples of facilities that would be available in Vicnet, extended from the small Victoria University campus community to a wider geographical area. They should give some idea of the power, versatility, and convenience which could be had from a public version of this small academic packet switching network.

5. CONCLUSION

What began as a simple desire on the part of three organisations at Victoria University for some common development in the area of data communications, has developed into an ambitious and productive foray into the field of packet switching, which could point some promising directions for the future of data communications in New Zealand. The strongest peg in the Vicnet argument is the standard called X.25: we can promise that users who work toward Vicnet will be working toward the future of data communications.

It is certain that mass data communication will grow in this country as it has overseas, that is, staggeringly, in the coming decade or so. It is extremely likely that much of the development will come in "soft" methods of data communication such as packet switching. We seriously hope that this work, done by a few people in an academic context, can be of help and of some modest influence on those in a position to engender larger, more practical, and possibly public packet switching networks in New Zealand.

Acknowledgements

The project, its antecedents, and its tributaries have involved many persons in several places over a period of more than two years. When we particularly thank Clive Nicholson, Mike Newbery and Chris Freyberg, we are restricting mention to those whose ideas have matured and been immortalised in the memories in the machines in Vicnet.

6. REFERENCES

[1] Comite Consultatif Internationale pour Telegraphie et Telephonie, "Recommendation X.25 for the interface between data terminal equipment and data circuit terminating equipment for terminals operating in the packet mode on public data networks". October 1976.

[2] Hopper K.R. "KiwiNet, an Experimental Computer Resource Sharing Network", Massey University Report No. 3, August 1976.

[3] International Standards Organisation, Draft International Standard 3309.2, Data Communication - High-level data link control procedures - Frame Structure", 12 June 1975.

[4] Bowles S.W. "Communication Protocols in Current Packet Switching Networks", Masters Thesis, Victoria University of Wellington, January 1977.

[5] McKendry M.S. "Proposal for a Virtual Terminal Protocol for Vicnet", Information Science Department Report, Victoria University of Wellington, October 1977.

[6] Hopper K.et al "KiwiNet: System Control and Access; Preliminary Specifications", Massey University Computer Centre Report No. 34, May 1977.

[7] Rybczynski A. and Weir D. "Questions and Answers on X.25 and Characteristics of Virtual Circuits for Datapac" General Note 146

of the International Network Working Group, February 1977.

[8] Brooks F.P. "The Mythical Man Month", Addison-Wesley 1975.

[9] Hirsch Phil "Protocol for Packet Networks: the Question is Implementation", Datamation, May 1976.

[10] Naylor T. et al "Vicnet Subscriber Software", Proceedings of this Conference.

[11] McKendry M.S. and Hine J.H. "Vicnet: Considerations in the Communications Subnetwork", Proceedings of this Conference.

****

VICNET:
AN X.25 IMPLEMENTATION FOR THE BURROUGHS B6700

AUTHORS: C.A. FREYBERG

Computer Unit
Massey University

M.D. NEWBERY

Computing Services Centre
Victoria University of Wellington

C.L. NICOLSON

Applied Mathematics Division
Department of Scientific and Industrial Research.

ABSTRACT:

The Victoria University of Wellington Campus computer network Vicnet relies on the Computing Services Centre's B6700 for several important host services. This paper is concerned with the interfacing of the B6700 to Vicnet using the international standard user interface X.25. The paper gives the necessary background and implementation of the software composing the X.25 interface.

KEY WORDS AND PHRASES:

Data communications, line protocols, X.25, HDLC, half duplex, asynchronous, packet, virtual terminal.

C.R. CATEGORIES:

3.81, 3.83, 3.89, 4.35, 4.39, 6.29

May, 1978

VICNET:
AN X.25 IMPLEMENTATION FOR THE BURROUGHS B6700

C.A. Freyberg

Computer Unit
Massey University

M.D. Newbery

Computing Services Centre
Victoria University of Wellington

C.L. Nicolson

Applied Mathematics Division
Department of Scientific and Industrial Research

ABSTRACT

The Victoria University of Wellington Campus computer network Vicnet relies on the Computing Services Centre's B6700 for several important host services. This paper is concerned with the interfacing of the B6700 to Vicnet using the international standard user interface X.25. The paper gives the necessary background to B6700 data communications and describes the design and implementation of the software composing the X.25 interface.

1. INTRODUCTION

The arrival of the B6700 on the Victoria University Campus in 1973, with its great potential for remote and interactive computing, stimulated an intense interest in the development of data communications on the campus. It was foreseen that a rather disorganised set of data communication links was likely to become established, such was the pressure for development.

In an effort to avoid unsatisfactory developments, a number of interested groups cooperated to lay down standards. This eventually led to the decision to create a packet switching network which would provide a wide variety of services generally available to all network subscribers.

This paper briefly outlines the historical events and decisions in the development of data communications in relation to the B6700. It then outlines the B6700 manufacturer's data communications system and software structure. Next, the decision to adopt the X.25 data communications standard is discussed and its current and planned implementation for the B6700 described.

In particular, the paper details and justifies the changes required to be made in the manufacturer's software to insert our software items which will eventually enable the B6700 to provide the most extensive host services available on Vicnet.

2. HISTORY

The arrival of the B6700 in 1973 followed by a few years the formalisation of packet switching network concepts. Both ARPANET [1] and ALOHA [2] had demonstrated the usefulness of this type of data communications.

Even before very many terminals had been connected to the B6700, arrangements were made to interface it to the Hawaii based ALOHA network via a satellite link. Although a large amount of preparatory work was done towards this, it was discovered that the (under-configured) B6700's Data Communications System was unable to carry out the functions required of it. The easiest solution to this problem was to "front end" the B6700's front end with a minicomputer. Before a suitable mini could be purchased, the satellite link became unavailable.

Following this, a project to link the Victoria and Massey University computers (both B6700s) was proposed and received funding [3].

In parallel with these developments another machine on the Victoria Campus had been connected to the B6700. This system, a Hewlett Packard HP2100, owned by Applied Mathematics Division DSIR, provided access to the B6700 or to its own Time Shared Basic service. It used a polled line discipline similar to that used by a number of "multi-dropped" Burroughs terminals on the B6700.

By the time the new front end (PDP 11/03) for the Victoria-Massey link (Kiwinet) project arrived, experience gained from the HP2100 - B6700 link had shown that a much better inter-machine line discipline was needed.

In order to link the PDP 11 and the B6700 a local discipline was grown and implemented while a more suitable discipline was sought. This search [4] led to X.25, which also provided the frame-work for all other planned services. Thus from late 1976 a number of implementers coordinated their efforts to provide compatible disciplines (from the lowest level upwards), currently involving 7 machines from 4 different manufacturers. During this time emphasis shifted from the Victoria-Massey link (Kiwinet) to the local Victoria Campus network (Vicnet).

Because the Victoria B6700 plays a major part in supplying computing services to Vicnet, there has been mounting pressure for its role in Vicnet to be enhanced to incorporate all network access procedures. These had earlier been thought to be extremely difficult to incorporate and to necessitate an additional front end. However, the hurdles which originally prompted us to front end the B6700's front end have been overcome by the software described in this paper.

3. DATA COMMUNICATIONS ON THE B6700

Data communications on the B6700 is handled via an independent Data Communications Processor (DCP) which controls the terminals and is responsible for the transfer of messages using appropriate line disciplines [5]. The DCP communicates, via direct memory access, with a portion of the Master Control Program (MCP) [6] running on the main system called the Data Communications Controller (DCC).

The DCC then routes messages to a program called a Message Control System (MCS). Each station is under the control of an MCS (note:- each MCS controls many stations) which is responsible for error recovery, signing users on to the system, switching messages and controlling the access of applications programs to the station.

Thus the structure of data communications on the B6700 is strictly hierarchical. Terminals connect to the DCP, which handles line protocols, translates to the B6700's internal character set (EBCDIC) and packages the input in a standard message format which it delivers to the DCC. The DCC forwards this message to the appropriate MCS which may then, depending on message type and content, route the message on to an application (via the DCC, which is the intermediary in all DCP/MCS/application transfers). The chain is reversed on output. See fig. 3.1.

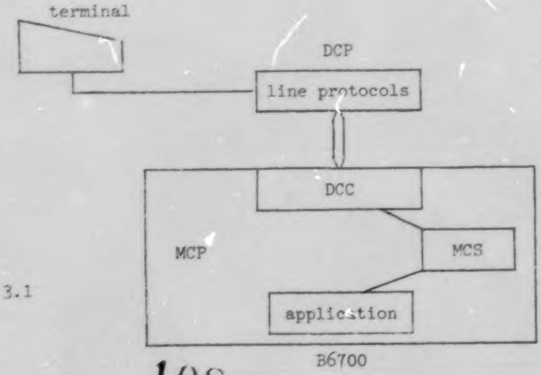


Fig. 3.1

Lower levels are effectively transparent to upper levels; e.g. the nature of the line discipline is invisible to the MCS and the identity and nature of the MCS is invisible to the application.

The format of the messages passed between the DCP, DCC and MCS is broadly as follows: -
A header, containing such information as message type, timestamp, station, etc. optionally followed by an information field. For instance, a text message (the most common type) would look roughly like this: -

TYPE = text	STATION	CARRIAGE CONTROL	TEXT LENGTH	OTHER FIELDS	T E X T

Fig. 3.2

Let us follow a line of text generated by an application program through the system.

```
          WRITE(1,99)
   99     FORMAT('ODOUBLE SPACED')                    Application
```

generates:

TEXT	STATION	space 2 lines before printing	13	...	DOUBLE SPACED

which is presented to the MCS, which forwards it through the DCC to the DCP, (assumed unedited) which prints on the terminal:

```
blank line
DOUBLE SPACED
```

Note that there is an implicit assumption that each text message constitutes one line on the terminal and that the DCP will add control characters to the text to accomplish the carriage control.

The DCP is programmed using Network Definition Language (NDL) [7], a high level descriptive language. The NDL compiler generates code for the DCP from a description of the terminal system provided.

4. EXISTING SERVICES

The B6700 system provides to the remote user all the facilities that the batch user has. These are maintained by the Remote Job Entry (RJE) software.

RJE operates a cluster comprising a line printer, card reader and operator's console. The cluster is extendable to include a card punch and terminals. The remote user also has access to a number of other control programs. These provide interactive editing and program execution (CANDE), APL - the well-known programming language (APL700), and a local control program for terminal control and limited program execution (INTERACT).

Other control programs are available but are not often used at Victoria - but, like most of the others, they enable programs to communicate with terminals as a basic function.

5. WHY X.25

X.25 is a data communications standard [8] which sets out a structured set of commands and responses to enable a subscriber to gain access to other subscribers via a communications network. This standard only details the interface to subscribers, hence implying that certain types of information must be conveyed across the network; it does not specify the internal workings of the network.

The X.25 standard is structured as a number of levels; each level is almost totally independent of the exact nature of the levels below it. We will explain their usefulness in the B6700 data communications context, starting at level 1 (the lowest level).

Level 1 of X.25 describes the structure of message frames and the generation of check characters, giving a means of conveying any number of bits over a link.

Level 2 uses a subset of the ISO HDLC [9] (High-level Data Link Control) procedures to establish a reliable error-free link. The ISO HDLC standard, which is also a subset of the ANSI ADCCP [10] (Advanced Data Link Control) standard, provided many of the features which we had found necessary for our own line disciplines. The extra procedures specified form a secure link level discipline. It is only in the levels above 2 that X.25 is any different from the other standards, and it is also here that the greatest gains in terms of access to network services are made.

The packet level protocols are very similar to the link level ones, so we will indicate briefly the philosophy behind them.

The packet protocols are based on the concept of a Virtual Circuit (VC) between subscribers. VC's may be permanently established or dynamically created via call set-up and termination procedures.

We concurred in the choice of X.25 as the network access protocol for Vicnet for the following main reason.

The ability to dynamically set up VC's is the greatest attraction to the use of X.25 within Vicnet. For the B6700 the various services available will be assigned "phone"numbers and any network subscriber will be able to place any number of calls to these numbers, without prejudicing any other subscriber's access to the same service.

6. HDLC

The B6700's makers have not yet distributed their version of HDLC for B6700s and there is no indication of this occurring in the near future.

Our implementation of X.25 on the B6700 will be outlined in the same order as it is in the X.25 report concluding with our local Virtual Terminal protocol implementation.

The ISO HDLC procedures are used to form the lowest level of X.25 on the B6700 system and their implementation is now described.

HDLC for asynchronous, half-duplex lines was implemented according to ISO draft standard ISO/DIS4335 except in being asynchronous rather than synchronous, and in omitting the SREJ frame (not used in X.25).

The discipline was implemented entirely within NDL on the B6700, without recourse to either special hardware or MCS assistance. Owing to the latter constraint the link level window size (the number of outstanding, unacknowledged messages permitted) is restricted to one. A single modification was made to the NDL compiler to produce code for calculating the HDLC FCS polynomial [11] rather than the normal Burroughs CRC.

The implementation is believed to be a complete implementation of the ISO HDLC standard with the omission of SREJ frames. In particular, all the unnumbered frames are handled. It is rather less strict than ISO over message sizes. It will transmit any message presented to it by the system, regardless of length. This may upset the recipient, which has the option of rejecting the message with a buffer size error (which is what the B6700 will do if it received an excessively long input message). In any case, the B6700 will not split a long message into several smaller ones.

Conventions Adopted to Achieve Asynchronous Operation

HDLC is defined only for synchronous transmission. The committee left asynchronous operation 'for further study'. Rather than insert a '0' after five contiguous '1's, which is both difficult and inappropriate to asynchronous operation (for instance, should the start/stop bits be included in the count?), the following convention is adopted: -

To achieve transparency of flags, an imbedded flag (01111110) is preceded by a DLE character (00010000). Any character following a DLE is not a flag and is to be allowed through. The DLE is discarded and does not participate in the formation of the FCS. It follows that any DLE character in the text must be preceded by a DLE.

The order of transmission of the bits within characters is deemed to be a function of the hardware in the asynchronous case: accordingly no attempt is made to alter the order of bits in the A or C fields for the purposes of FCS calculation. (The ISO standard defines the least and most significant bits in these fields, which is sufficient for our purposes.)

The ISO standard distinguishes between a 'primary' and 'secondary' (DCE and DTE respectively in X.25 nomenclature, approximately 'host' and 'terminal' in common parlance). Now the ultimate linkage is DTE - DCE - DTE, where the internal nature of the DCE is undefined and in the degenerate case is null. Accordingly, we allow either (or both) parties to consider themselves the primary, and to set the poll/final bit to request immediate acknowledgement of outstanding messages. The B6700 uses this bit to turn the line around and reduce message crashes on a half-duplex line. This use does not appear to conflict with the standard and the B6700 will function correctly if its partner does not apply this convention. (If the partner does not set the bit, the B6700 will wait some time after a message for further input before transmitting. If the partner misuses the bit and transmits a frame immediately following a frame with the bit set, a message crash may result.)

Implementing HDLC Within the Constraints of the B6700 System

Line disciplines on the B6700 have to be written in NDL within certain rigid constraints.

The operating system which resides in the DCP provides a number of routines to carry out common data communications functions. These functions are generally independent of the characteristics of the devices connected to the DCP, but use values from tables to handle devices' peculiarities. The operating system also maintains the scheduling of the DCP processes which control the lines (control sections).

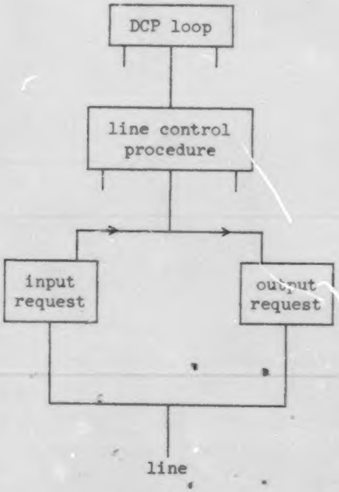


Fig. 6.1

Each line has specified for it a line control procedure and each station on a line has both an input and output procedure (Input and Output Request sets). A full-duplex line (2 lines) has two possibly different line control routines. There is no need for all stations on a line to have the same input/output procedures, but in practice they do. The major difference between the control sections and the Request sets is: - the ability to get and return messages from the main system; the ability of the control section to call the necessary Request sets depending on the current station (Station Index) for the line in question.

The variables available to the NDL program are limited in number and scope, and are of the following classes: -

Global - system status such as NOSPACE, DUMPING.

Line - can be seen by all procedures on a given line; includes such status as BUSY, QUEUED and user allocated variables.

Station - can be seen by all procedures when the STATION INDEX is set to a given station. These include user defined variables, and a number of system defined variables such as FREQUENCY, READY.

Message - can be seen only by the Request set.

Until recently the number of user defined variables available was severely limited. This made it very difficult to implement anything but standard line disciplines and HDLC nearly impossible.

Within the control section are handled all the HDLC frame types which do not carry information fields. Incoming information frames are also handled there when it is impossible to get into the Input Request set. This state will exist when the main system has made a station disabled for input, or generally not ready.

The Output Request set only transmits frames containing text originating on the main system. The Input Request set passes on incoming information frames to the main system when first correctly received.

In the control section we have had to duplicate some functions also contained within the Request sets, due to NDL imposed constraints.

Interim Virtual Terminal Implementation

In the current implementation it has been found necessary to insert the high-level virtual terminal protocol [12] in the HDLC implementation. Hence we briefly describe it and its effect at this stage, reserving full discussion until a more appropriate part in the paper.

As mentioned above, the message presented to the DCP for output has carriage control fields associated with it. We therefore edit the message at this point before passing it on. We cast the text into the Virtual Terminal form. Virtual Terminal is a level above packeting (level 3 of X.25) and is done in the DCP only because level 3 is as yet unimplemented. With the addition of packeting this will have to be shifted (unless packeting too were to be done by the DCP, which is considered impractical).

Current Situation

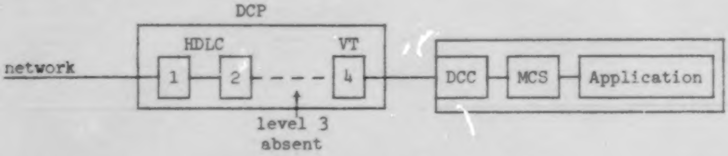
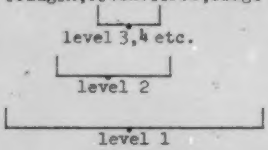


Fig. 6.2

Application	
to MCS	[text]
to VT	[MESSAGE[text]]
to HDLC level 2	[text in VT form]
to HDLC level 1	[A,C[text]]
to network	[flag[A,C[text]]FCS,flag]



7. PACKET LEVEL X.25 ON THE B6700

Level 3 of X.25, the packet level, provides a means of delivering messages to subscribers anywhere on the network. Each subscriber has a unique identifier/number exactly analogous to a telephone number. Note that subscriber here means whatever is identified by the phone number. One installation might have several numbers, each allocated to different services. One might call one number for RJE, another to transfer files, etc.

A message may be split across several packets, or contained in one packet (but one packet will not contain more than one message). Thus there is no restriction on the size of messages carried by the network although packet size is restricted.

Examples of messages packeted by X.25: -

(a) [Format, address, P(R), 0, P(S), 0 [msg1]]

(b) [Format, address, P(R), 1, P(S), 0 [msg2...]]
[Format, address, P(R), 0, P(S), 0 [...msg2 continued]] .

Level 3 of X.25 presented a problem on the B6700, namely, where to implement it. It is rather more complex than can be conveniently described in NDL and requires information and decision making not strictly appropriate to the DCP.

On consideration, the proper place is in an MCS, as shown below.

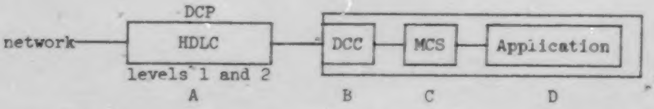


Fig. 7.1

Unfortunately a problem now appears. Traditionally MCS's on the B6700 have not been confined to reformatting messages for applications but have performed as applications themselves. Examples of MCS's are APL, RJE and CANDE (Command AND Edit language). So the packet switching would need to be duplicated in all such MCS's to which network access is provided, This is undesirable. Therefore we can duplicate the packet switching functions, deny network access to other MCS's, or find somewhere else to put the packet switching.

Obviously, the correct place for the packet switching is somewhere between B and C in fig. 7.1. We have, in fact, begun the implementation of a means whereby an MCS may insert itself between a station and the nominal controlling MCS; i.e. we allow an MCS to monitor stations. It passes nominal control of the station to another MCS, forcing all traffic for the station to pass through the monitoring MCS. Such an MCS may of course function in the normal way.

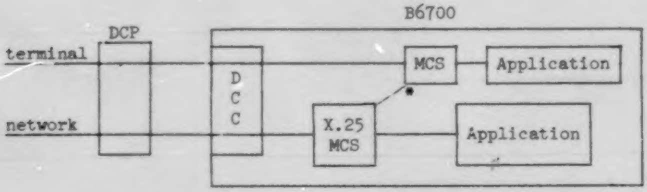

B6700
DCP
terminal
network
D
C
C
MCS
Application
*
X.25
MCS
Application

Fig. 7.2

The linkage marked * in fig. 7.2 is achieved by means of a dummy station. It may be noted in passing that the same effect could be obtained by physically connecting two lines back-to-back thus: -

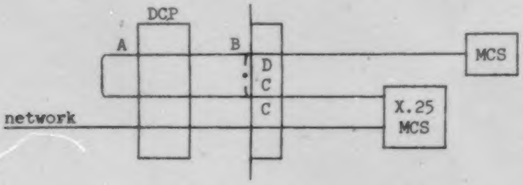

DCP
A
B
D
C
C
MCS
network
X.25
MCS

Fig. 7.3

All we are doing is moving the short circuit from A to B and incidentally saving two lines in the process.

If we now follow a message through the system we see: -

	application	[text]
to	X.25 MCS	[MESSAGE [text]]
to	HDLC level 2	one or more [MESSAGE[packet header[text]]]
to	HDLC level 1	[A,C[packet header[text]]]
to	network	[flag[A,C[packet header[text]]]FCS,flag]

The facilities provided by the X.25 MCS are as follows: -

7.1 An application attempting to communicate with an X.25 line will first engage in a dialogue with the MCS to indicate which number it wants. The MCS will then request the call (or select the appropriate Permanent Virtual Circuit, or clear the call). Thereafter the MCS performs the packeting functions for the application until call termination.

7.2 A non-X.25 station, i.e. one not on the 'network' line whose control is passed to the MCS will be treated like an application program.

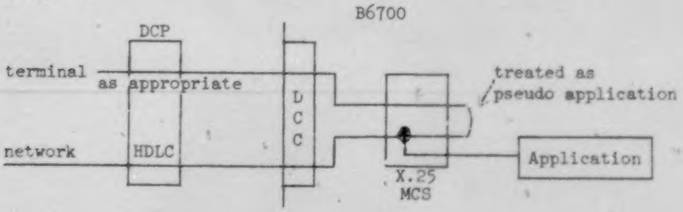

B6700
DCP
terminal
as appropriate
D
C
C
treated as
pseudo application
network
HDLC
X.25
MCS
Application

Fig. 7.4

7.3 An incoming call from the network will be handled as follows: -

7.3.1 The call may be to an application supported by the X.25 MCS, for instance a 'file transfer' number. The MCS then initiates the application

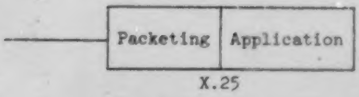

Packeting
Application
X.25

Fig. 7.5

The situation after initiation is identical with that of cases 7.1 and 7.2 above.

7.3.2 The call is destined for some other MCS.

(Note: Each MCS to which network access is permitted will be assigned its own 'phone number'.)
In this case a 'spare' station is obtained from a pool of dummy stations maintained by the system and nominal control of the station passed to the required MCS.

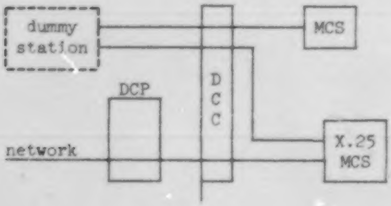

dummy station
MCS
DCP
D C C
network
X.25 MCS

Fig. 7.6

The situation on the B6700 may therefore be represented thus: -

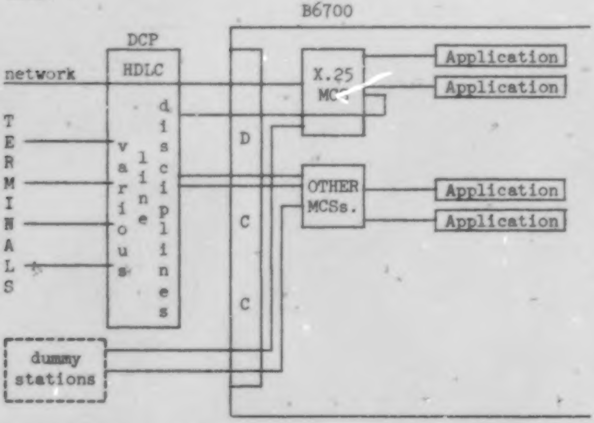

B6700
DCP
HDLC
network
TERMINALS
various
line
disciplines
D C C
X.25 MCS
OTHER MCSs.
Application
Application
Application
Application
dummy stations

Fig. 7.7

8. THE IMPLEMENTATION OF THE VIRTUAL TERMINAL

For terminal application interfacing we have implemented the Virtual Terminal protocol (henceforth shortened to VT). This exists at a level above packeting and is used to present messages to generalised terminals. (N.B. VT is fully independent of the levels below it and it is quite feasible and even useful for processes to converse using VT; e.g. an input process, on receipt of a SEND command, such as a CR character, passes a NEWLINE message to the output process which echoes a new line on the terminal.)

VT gives the application back the 'feel' of the terminal which has been buried by levels of packeting, switching and so on. It also, by providing blank compression, allows the message which has swollen somewhat, to be shortened.

At the VT level one has a means of specifying commands to a generalised terminal and of interrogating the precise capabilities of the terminal.

As is mentioned above, one of the fields in a text message (in 'DCC' format) is devoted to carriage control and this information is extracted and used to generate a VT message from the initial text. Thus VT on the B6700 is transparent to the applications at this stage. Such is not generally the case. If a system provides a general method of terminal control it may be converted to VT form, otherwise the application has to generate VT.

Currently the VT is preduced by the DCP (see above), but with the introduction of the packeting MCS this will change as a higher level cannot be introduced into a lower one.

VT will become a function of the packeting MCS, applied prior to level 3 of X.25.

So, let us once more follow the progress of a message through the system.

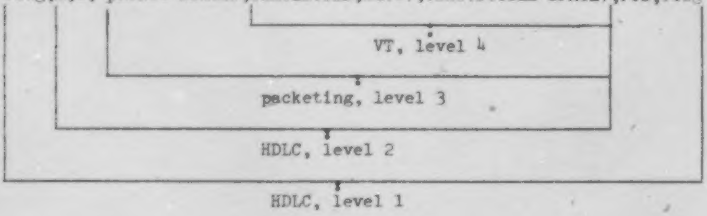
application	[text]
to MCS:VT	[MESSAGE [text]]
to packeting	[MESSAGE [text in VT]]
to DCP:HDLC level 2	one or more [packet header[text in VT]]
to HDLC level 1	[A,C,[packet header[text in VT]]]
to network	[flag,[A,C,[packet header[text in VT]]]FCS,flag]

If we take the message generated by the FORTRAN fragment shown previously we might have

```
      WRITE(1,99)
 99   FORMAT('ODOUBLE SPACE')                          Application
 space 2 before printing [DOUBLE SPACE]                to MCS:VT
 [TEXTLETTER,NEWLINE(2),TEXT(DOUBLE SPACE)]            to MCS:packeting
 packet header,[TEXTLETTER,NEWLINE(2),TEXT(DOUBLE SPACE)]  to DCP
```

etc., until finally

flag,A,C, packet header,TEXTLETTER,NL(2),TEXT(DOUBLE SPACE),FCS,flag



9. TESTING

Because of the complex nature of a complete X.25 implementation and the number of different machine types involved, testing could have been a 'factorial' type problem. This situation was avoided by carefully staged implementation and extensive self-testing at each step.

All implementers involved agreed, on a number of occasions, to implementation levels. This meant that each time machines were 'plugged' together they could function completely with the subset of the functions agreed upon.

Before inter-machine testing was started, extensive intra-machine testing was carried out by performing back-to-back tests. Although this method does not guarantee that the systems are compatible, it did allow the detection of faults which otherwise would have been difficult to diagnose.

On all testing soft line monitoring was carried out to insure that line data was as expected. Implementation and testing of the B6700 NDL was carried out at both Massey and Victoria at times independently and simultaneously, for reasons of convenience.

10. SUMMARY

We have described the implementation of an X.25 interface on the B6700, and an outline of network services as we envisage them developing.

Integrated circuits to perform the lower level functions (level 1 of HDLC) are becoming available and would enable the HDLC software to be made much simpler, as well as giving us true synchronous bit stuffing HDLC.

Vicnet is not merely an end in itself; it will serve as the basis for further resource sharing by the parties involved.

To date we have experienced major gains in reliability and efficiency from the use of the link level protocols alone and the utilisation of the Virtual Terminal discipline has rationalised our approach to terminal interfacing, with consequent gains in ease of terminal control and use.

Acknowledgements

We should like to acknowledge the cooperation and contributions of the following to this project: - S. Bowles, J. Hine, M. McKendry, T. Naylor, J. Rumsey, D. White and P. Williams.

11. REFERENCES

[1] Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R., Walden, D.C. The interface message processor for the ARPA computer network. AFIFS SJCC 36 (1970), pp. 551-567.

[2] Abramson, N. The ALOHA System. Computer communications networks. Prentice-Hall. pp. 501 (1973).

[3] Hopper, K.R. Kiwinet, an experimental computer resource sharing network. Massey University Report No. 3 (Aug. 1976).

[4] Bowles, S.W. Communication protocols in current packet switching networks. Masters thesis. Victoria University of Wellington. January 1977.

[5] Burroughs. B6700/B7700 Data communications functional description. 5000 060.

[6] Burroughs. Master Control Program: Information Manual. 5000 086, 1970.

[7] Burroughs. Network Definition Language: Information Manual. 5000 078, 1971.

[8] Comité Consultatif Internationale pour Telepgraphie et Telephonie. Recommendation X.25 for the interface between data terminal equipment and data circuit terminating equipment for terminals operating in the packet mode on public data networks. October 1976.

124

[9] International Organisation for Standardisation Data communication - High level data link control procedures ISO/DIS 4335. 1976.

[10] American National Standards Institute Advanced Data Communication Control Procedures (ADCCP) (6th Draft Proposed).

[11] Nicolson, C.L. Parallel calculation of cyclic redundancy check characters. Unpublished report, Applied Mathematics Division, Department of Scientific and Industrial Research, N.Z. 1977.

[12] McKendry, M.S., Williams, P.R. Proposal for a Virtual Terminal Protocol for Vicnet. Information Science Department Report, Victoria University of Wellington. (Aug. 1977).

VICNET:

CONSIDERATIONS IN THE COMMUNCIATIONS SUBNETWORK

AUTHORS: * M.S. McKENDRY
H.H. HINE

Department of Information Science
Victoria University of Wellington

ABSTRACT:

The Communications Subnetwork in an X.25 packet switching network is a distinct entity which forms the network's basis. This paper describes the considerations and pitfalls in the design of the subnetwork for Vicnet.

KEYWORDS AND PHRASES:

X.25, end-node, network, subnetwork, ARPANET, virtual circuit.

C.R. CATEGORIES:

3.81, 3.83, 3.89, 4.39, 6.29

---

* formerly Applied Mathematics Division, D.S.I.R.

VICNET:
CONSIDERATIONS IN THE COMMUNICATIONS SUBNETWORK

*M.S. McKendry
J.H. Hine

Department of Information Science
Victoria University of Wellington

ABSTRACT

The Communications Subnetwork in an X.25 packet switching network is a distinct entity which forms the network's basis. This paper describes the considerations and pitfalls in the design of the subnetwork for Vicnet.

INTRODUCTION

Vicnet is a prototype data communications network being developed at Victoria University as a joint effort by the Department of Information Science, Computer Services Centre and the Applied Mathematics Division of D.S.I.R.[1]. At an early stage it was decided to design the network using the X.25 specification for the interface between the network and its subscribers [2]. The X.25 specification provides a basis for the standardisation of network interfaces. It specifies the protocol by which a subscriber communicates with the end-node (fig. 1), but does not go beyond this. Just as any computer may be a subscriber provided it presents an X.25 interface to the network, so any network may be an "X.25 network" provided it presents an X.25 end-node interface to the subscriber. The term "network" will be used hereafter to mean a totally X.25 network; one which only has X.25 interfaces, such as Vicnet.

Figure 1 shows an X.25 network: its boundary is crossed only by links which conform to the X.25 standard. This constrains

---

* formerly Applied Mathematics Division, D.S.I.R.

.../2

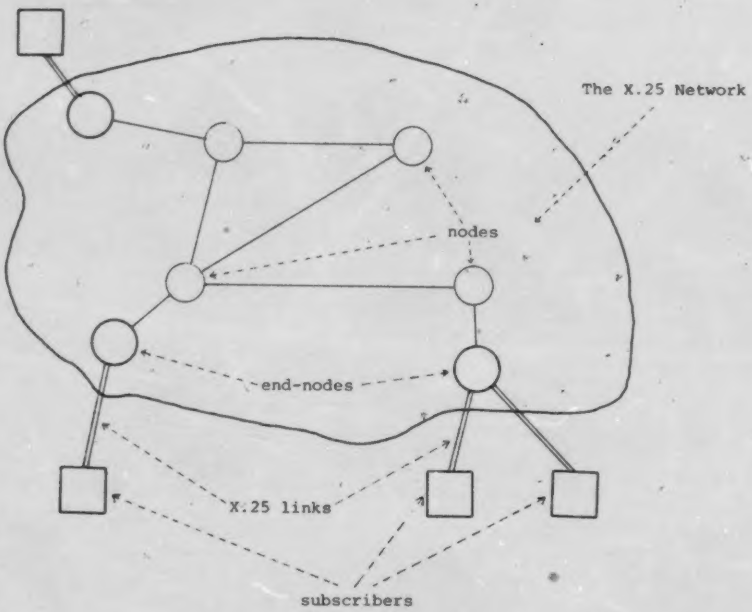


FIGURE 1

the software in the end-nodes somewhat, since it must present an X.25 protocol to the subscriber. However, we are unconstrained beyond the software which supplies this interface. X.25 does not specify how the inner network functions.

The "unconstrained" part of the network is known as the "communication subnetwork" or "subnet". While X.25 does not indicate how the subnet must function it does specify the services which must be provided and thus gives us many clues as to how the X.25 designers viewed the subnet. The network (and, perforce, the subnetwork) must be able to establish calls upon receipt of an X.25 call request packet at an end-node, and it must be able to clear calls and carry out other such "housekeeping" functions as are defined in the X.25 protocol. It must attempt to

.../3

deliver all data packets sent into the network to the appropriate party in the correct order, without loss or duplication of information or data, and within a (small) finite time.

These services could easily be handled if the nodes had infinite speed, memory capacity and never failed. Unfortunately this is far from true, and we find ourselves in a familiar situation: the communications subnetwork is a finite resource which must be shared equitably (or at least deterministically) amongst all those users who are making demands upon it.

This paper addresses the design of a suitable subnetwork, for Vicnet, which we hope achieves the stated goals.

DESIGN CONSTRAINTS

Vicnet is designed in several levels [1]. At the lowest levels the character by character transmission between nodes is implemented. Each successive level is built on top of the previous levels and provides a more sophisticated service. The development of the lower levels is relatively straight forward and most often dictated by the technology used. The level of the subnetwork which controls the competition for resources by the network users, while not being entirely independent of the lower levels on which it is implemented, may be isolated from many components of the network logic. The choice of algorithms to use at this level is still very much an open question.

There are two major considerations. First we consider the congestion problem. If all network users were to attempt to push the maximum traffic through the network at one time the subnetwork may become so congested that nothing will be able to move. This situation is a deadlock. A common form of it is the store-and-forward deadlock in which two adjacent nodes are full up with packets bound for one another. Neither mode can transmit because his partner has no space for any incoming packets; neither node can make space for incoming packets because they cannot transmit in order to free space. Approaches to this problem and the associated deadlock resolution or avoidance fall under the general

.../4

title "flow control".

Flow control is closely related to the second major consideration, the routing algorithm. The routing algorithm is that part of the network logic which determines which path a particular packet will take through the communications subnetwork. The algorithm attempts to maintain a predefined minimum service level (an upper bound on allowable transit delay) regardless of node, end-node or line failure. The designer hopes that it will assist us to maximise throughput and minimise transit delay, although often these must be traded off one against another. Obviously a good routing algorithm should be able to prevent local congestion by routing incoming traffic along a different path. However, we cannot expect a routing algorithm to prevent congestion when the total network is overloaded. This is the responsibility of the flow control procedure.

We will consider two main approaches to subnetwork design and examine their respective implications and merits with regard to the two major problems: flow control and routing. The first, and more common approach, is the datagram subnetwork. In the datagram approach the path of each packet through the subnet is determined independently as the packet enters or traverses the communications subnetwork. [ARPA].

The second approach to subnetwork design is the virtual circuit subnetwork. This requires nodes to "remember" all calls which pass through them. Once a call is established all packets follow the same path [GMD]. The virtual circuit approach is comparatively new but has distinct advantages which have led to recent gains in its popularity, particularly for X.25 networks since it reflects the virtual circuit approach which is already established from subscriber to subscriber.

Note that while we must maintain a virtual circuit between subscribers we are not obliged to use virtual circuits in the subnet. We must merely maintain the illusion of a virtual circuit between subscribers.

DATAGRAM SUBNETWORK ARCHITECTURE

The main benefits of the datagram approach lie in the flexibility which

.../5

accrues as a result of the dynamic route selection for each packet. The subnetwork can adapt rapidly and easily to any environmental changes, such as congestion or node failure. Incoming traffic is simply routed around the affected area until the trouble clears up.

The choice of a route selection mechanism has entertained many minds over recent years. Firstly we must eliminate the obvious mechanism, a global intelligence which oversees the entire network and routes packets optimally through it. Apart from the delays involved (interrogating the intelligence and awaiting a reply will double average subnetwork transit time), there is the possibility that the network will be severed, resulting in one part of the network having no brain.

We are led to the conclusion that some form of distributed routing intelligence is required. We will now examine the proposed system for Vicnet, the concept of which is already embodied in the ARPA network [3].

Each node maintains two tables. The first, the delay estimates table, holds a value which is an estimate of the transmission delay to each possible destination for each line leaving the node. For a given destination the optimal route leaves the node via the line with the minimum delay estimates for that destination. The optimal routes are held in a second table, the routing table, which contains a single entry for each possible destination. The routing table can obviously be constructed from the delay estimates table.

The delay estimates table is updated periodically with information from neighbouring nodes. Two adjacent nodes exchange copies of their delay estimates table. The entries corresponding to the line joining the nodes can be updated by taking the entry from the neighbour, adding 1 for the transmission delay to that neighbour and adding a value which is the function of the number of packets currently queued to go out that line. This becomes the new entry in the delay estimates table. The routing table can be updated from the delay estimates table. In this manner the information propagates from node to node, eventually reaching the farthest reaches of the network.

.../6

This routing algorithm, while being reasonably simple to implement, does have distinct disadvantages. The main one of these is the propagation time of the news of changes through the subnetwork. This arises because news only travels one link each time estimates are exchanged. To reduce this propagation time means increasing the frequency of estimate exhanges, which in itself increases delay times. In ARPANET estimates are exchanged every 0.2 seconds over lines which run at 50,000 bps.

Interestingly, the propagation time for bad news is higher than that for good news. Since each node selects the best estimate they tend to disbelieve high estimates which arrive for a previously optimal path, choosing instead to accept a lower estimate from another node. This lower estimate may be dependent on the same broken link which caused the previously optimal path to be discarded, but the news of the break has not yet propagated as far as the node making the accepted estimate.

An important cost of the datagram approach is the "address" which each packet must carry. Since the nodes have no knowledge of individual calls it is the responsibility of the transmitting end-node to add on to each packet an address indicating where the packet is going and who sent it. This is removed by the receiving end-node before the packet is forwarded to the subscriber and so is invisible to the subscribers. The addition of even a sixteen bit address field may have a significant effect on network overheads.

A further cost arises as a result of the dynamic routing of packets. Since two packets on the same call may take different routes through the subnetwork they may arrive in any order at the receiving end-node. The end-node must have sufficient storage to hold outstanding packets on the call so that they may be delivered to the subscriber in the correct order. This is required by the virtual circuit which X.25 uses to implement calls between subscribers. The amount of storage required may be reduced by reducing the frequency of route changes at the nodes which will lower the likelihood of packets arriving out of order. The frequency of route changes and its associated flexibility was one of the reasons for selecting a datagram subnetwork though, and it would be a pity to discard this feature.

.../7

However, even if we do allow sufficient space for reordering or somehow eliminate the reordering problem, we can can still end up with a packet attempting to enter a full end-node. If we do not control the entry of packets into the net this will eventuate in any network.

X.25 provides a means of slowing or even stopping a subscriber from sending packets into the network. When the receiving end-node gets near full it must somehow invoke this mechanism or the resulting back pressure will increase the likelihood of deadlock. All networks are faced with this possibility.

Solutions range from having the transmitting end-node actually request space (by sending a packet and awaiting reply) at the receiving end-node (ARPA) to more sophisticated systems based on classes of buffers at all nodes along the route of a packet (GMDNET). Unless we actually accept the reservation of space at the packets destination (which inreases traffic and transit delay) we must adopt a heuristic solution which will work most of the time, otherwise relying on back pressure as packets build up in the nodes approaching the affected end-node.

VIRTUAL CIRCUIT SUBNETWORK ARCHITECTURE

In a virtual circuit subnetwork a path through the subnetwork is established when the subscriber sends a call request packet to his end-node. This path, which must be remembered by all the nodes on it, is then used for all packets during the call.

The first and most obvious disadvantage of the approach is that each node must be able to recognise a call-request packet and "remember" the call. Associated with this is logic to clear the call. Both the code to do this and the data space required increase the size of the node machines. Since the memory of a node is finite there will also be some limit on the number of calls which may be in progress through any node at one time.

In its favour, the virtual circuit approach eliminates any need for packets to carry an address, as this is implicit in the route upon which

.../8

the packet is travelling.

Because routing is carried out only once per call we can spend more time determining an optimal path. It becomes possible to route high traffic calls (file transfers) through less congested areas and lower traffic (interactive) calls along the fastest path. These decisions must be made carefully in any case, since once the call is established it is difficult to change its route.

The cost of maintaining service for calls which pass through a failed node is high. To reroute a call while maintaining the correct ordering of packets in transit requires logic almost as complex as that required for an entire datagram subnetwork. This increase in software cost may not be justifiable.

It is in a stable running state that we see the main advantage of a virtual circuit subnetwork. Because all packets follow the same path they will arrive at the destination end-node in the correct order. With information about calls as they are established we can select optimal routes for each call, allowing us to spread the load through the entire subnetwork. Thus a virtual circuit subnetwork can handle higher loads more easily, provided there are no failures.

CONCLUSION

Comparing the two subnetwork architecture and relating them to the New Zealand situation we see that:

a) The datagram subnetwork adapts more readily to pathalogical situations, while the virtual circuit approach shows its strength in a free-running state.

b) The datagram subnetwork requires additional information (the address) to be attached to each packet, increasing line overhead. The virtual circuit scheme does not require this address, but does require the nodes to remember calls. This places a limit on the number of calls through a node.

.../9

c) In order to handle failures the virtual circuit approach requires logic almost as complex as the datagram approach. Additionally it requires logic in the nodes to remember calls.

While no final choice of network architecture has yet been made for Vicnet, our tendency is toward the datagram approach. While this is not in the spirit of X.25 virtual circuits it is compatible with them, and will later be able to handle subscriber-to-subscriber datagrams should this be required.

We believe that the New Zealand situation does not warrant the added complexity of the virtual circuit structure. New Zealand is a long country and it is unlikely that there will ever be more than one possible route between two N.Z. Universities.

An over-riding consideration in this decision will be the logistic effort of implementing a complex node. Software costs are high and, with the scarcity of qualified personnel, it is difficult to justify added complication unless it is clearly warranted.

REFERENCES

[1] Bowles S.W. "Vicnet, an X.25 Packet Switching Network", Proceedings of this Conference.

[2] CCITT, "Recommendation X.25 for the Interface Between Data Terminal Equipment and Data Circuit Terminating Equipment", October 1976.

[3] McQuilland J.M. and Walden D.C. "The ARPA Network Design Decisions", Computer Networks Vol 1,5. August 1977.

[4] Hermann J "Flow Control in the ARPA Network", Computer Networks Vol 1,1. June 1976.

[5] Kimbleton S.R. and Schneider G.M. "Computer Communication Networks: Approaches, Objectives and Performance Considerations". Computing Surveys Vol 7,3. September 1975.

.../10

[6] Heart F.E. et al "The Interface Message Processor for the ARPA Computer Network", AFIPS Spring Joint Computer Conference, 1970, Vol 36 pp 551-567.

[7] Raubold E. "GMD NETZ: Goals and Structure", Internal Document: Gesellschaft fur Mathematik und Datenverarbeitung, Institut fur Datenverarbeitung, Darmstadt, Federal Republic of Germany, October 1975.

****

VICNET SUBSCRIBER SOFTWARE

AUTHOR T.H. Naylor

Computing Services Centre,
Victoria University of Wellington.

S.W. Bowles
* M.S. McKendry,

Information Science Department
Victoria University of Wellington.

ABSTRACT

In a packet switching network there are essentially two different classes of computer: the 'network' machines and the 'subscriber' machines.

This paper describes the protocols which the subscribers must observe and the constraints this places on their software. The paper concludes with a case study of the software used to interface a micro processor to Vicnet as a terminal interface processor.

KEY WORDS AND PHRASES

X.25, link protocol, packet protocol, subscriber, terminal interface processor, assembler.

C.R. CATEGORIES

2.19, 3.81, 3.83, 3.89, 4.11, 4.13, 4.35.

---

* Formerly Applied Mathematics Division, Department of Scientific and Industrial Research.

INTRODUCTION

Vicnet is an experimental packet switching network which is being implemented to conform with the X25 (1) standard at Victoria University. The motivation and philosophy behind Vicnet is described elsewhere (3).

The part of Vicnet we chose to emphasise in our initial implementation effort was the user interface; and we decided that this would be as close to the standard CCITT specification as our powers of interpretation would allow. Correspondence with other organisations in North America and Europe indicates that our thinking is in line with, and of a level comparable to, organisations implementing X25 elsewhere in the world.

We attempt here to describe the X25 design, and to indicate briefly where our implementation has had to bow to local constraints.

LINK PROTOCOL

There is nothing terribly interesting about the link level protocol; it's just HDLC, which might be described as the culmination and climax of the last decade's research and experience in line disciplines. It governs physical transmission, with error detection and recovery, and further allows the line to be broken down into several independent "stations" for multidropping - though this last feature is not used in X25.

The link protocol itself may be split into two levels. At level one, we define a block or "frame" consisting of a beginning and ending character of binary 01111110 (called a "flag"). Between flags, we have a variable number of bits; the first two characters and the last two have special meanings, and the rest is the "information" part of the frame. The information field may contain any number of bits, not necessarily modulo eight, nor need the bits necessarily represent characters. To maintain objectivity, the X25 spec calls eight-bit groups "octets". Note that the use of special octets of value 01111110 to initiate and terminate the physical frame implies that we must adopt a convention to allow the user to send that particular sequence within his information field if he wishes; this function is referred to as "transparency" and is achieved by the transmitter's detecting any sequence of five consecutive "1" bits in the output stream and inserting a dummy "0" bit thereafter; the receiver reverses this logic.

The last two octets in the frame follow the user's information and are called the "frame check sequence". They constitute the sixteen bit "cyclic redundancy check" function of the bits of the frame, and are calculated by the sender and checked by the receiver to enable detection of physical errors. The first two octets in the frame precede the user's information and are called the Address or A octet and the Control or C octet. The A field is used for multidropping in HDLC and is effectively not used in X25. The C field is used for sequencing transmissions and acknowledging receptions. See Figure 1.

This leads us into the second sublevel of the Link Protocol, wherein physical errors are detected and ligical syncrhonisation established and maintained between the two machines. This is all done nicely using only the C octet, which splits down as in Figure 2. The most important fields are the three bit Send Sequence Number N(S) and the Receive Sequence Number N(R). The N(S) is bumped up by one, modulo eight of course,

Address "A"	Control "C"	Packet, 0 to 1048 bits, present only in "I" type frames

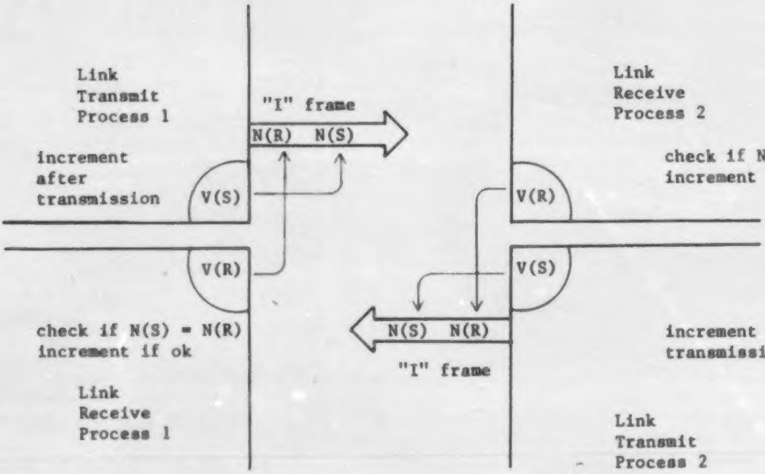
(transparency and redundancy inserted at a lower level)

(a) X25 "Frame" Format

--------> direction of transmission

	8	7	6	5	4	3	2	1
I frame	N(R)			P/F	N(S)			0
S frame	N(R)			P/F	S		0	1
U frame	M			P/F	U		1	1

(b) X25 Control (C) Field


Link
Transmit
Process 1
"I" frame
N(R) N(S)
increment
after
transmission
V(S)
V(R)
Link
Receive
Process 2
check if N(S) = V(R)
increment if ok
V(R)
V(S)
check if N(S) = N(R)
increment if ok
N(S) N(R)
"I" frame
increment after
transmission
Link
Receive
Process 1
Link
Transmit
Process 2

(c) X25 Link Level Sequencing Mechanism

FIGURE 1: X25 LINK PROTOCOL

on each successive transmission from an X25 machine; the N(R), then, is set to indicate which sequenced frames our machine has correctly received from the other machine. The device is simple; if N(R)=6, say, on a received frame we may take this to mean that our correspondent has received successfully all our frames up to but not including number 6 (the frame with N(S)=6, that is). At the time the discipline is established between the two machines, we agree on the number of frames which can be sent from machine A without any acknowledgement from B; this parameter of the line is called k, the "window size".

Thus acknowledgements are normally "piggybacked" on information frames; but there do exist other types of frames in which we can specify explicit acknowledgements and other control functions. Up to now all the frames we have discussed had a Control octet with a zero in the low order bit, which made them "informing frames"; a one in the low order bit, though, declares the frame to be of either Supervisory or Unnumbered type - depending, in fact, on the next-higher bit. An example of a Supervisory frame is the RR or Receiver Ready, which corresponds to the simple, explicit, un-piggybacked acknowledgement; Unnumbered frames, like Set Asynchronous Response Mode or SARM, are used to initialise the line, disconnect the line, report protocol errors, and other high level line control functions.

The present Vicnet implementation is a logical subset of all this, which will be expanded to meet growing needs as time goes on.

At level one, we have had to make some concessions to our essentially asynchronous transmission hardware. Our frames must all be an integral number of octets; and "transparency" is achieved, byte-oriented, by sending an Ascii "DLE" octet (00010000) to precede any octet of the information field which happens to look like a flag (01111110) - or like a DLE.

At level two, we have implemented (to name the technical names) Information frames, RR and REJ Supervisory frames, and the SARM, DISC, and UA Unnumbered frames. We have not yet included the RNR supervisory frame, nor the rest of the unnumbered frames. The window size has been left at one to simplify initial experiments; and in fact no piggybacked acknowledgements are currently being sent, though they will be properly processed if received.

PACKET PROTOCOL

What's new and exciting about the CCITT's X25 is that the third, or packet level, whose constructs ride within the line discipline already described. The standard is controversial here both technically and politically - but not less useful, because it gives us a secure base on which to build a standard, and maybe eventually a public, packet switching network.

The reason X25 can be seen as "politically" controversial is that it advances the view of the common carriers (who control the CCITT) that the basic mode of service to be rendered by a p.s.n. should be the "virtual circuit". This idea is simple - being completely analogous to a telephone call; and therein lie its strengths and its weaknesses.

So X25 assumes that a network subscriber will "ring up" another network subscriber, "speak" for a while, and then "hang up" on the call. It thus defines the formats of special packets for "call request" and "call clear", as well as the garden variety "data transfer" packets which may cross the network after the call setup and before the call clear. Packets are symmetrically defined to allow a subscriber to receive an "incoming call" or a "clear indication" passively from somewhere else in the network. This whole subtype of the basic virtual circuit service specified by X25 is called the "virtual call", VC, or (only occasionally) the "Switched virtual circuit". See Figure 2.

The other subtype is really a special case of the virtual call (VC). It's called a "permanent virtual circuit" (PVC) and is exactly like the virtual call except that, effectively, the network makes the call request for you, to a particular destination subscriber, once and for all when you sign up for the p.s.n. It's just like a telephone "hot line"; whenever the subscriber machine sends a data transfer packet in, it goes to that same prespecified destination subscriber.

A given subscriber may wish to have more than one (normally completely independent) VC or PVC in progress at any one time; to this end a "logical channel" field is provided in the packet header. It's just a reference number between the subscriber and the network, and each logical channel is handled by the net completely independently. Thus you may put in a call request to a branch office in Dunedin on channel 13 of your line to the network, while carrying another call on logical channel 26, even as you are transferring data packets down a PVC using channel 7.

Also provided within the data transfer packet are fields called the "packet send sequence number" or P(S) and the "packet receive sequence number" or P(R), which provide a means of packet sequencing and "flow control". The subscriber machine increments the send sequence number on each successive packet it transmits into the network, and conversly expects each packet it receives to show a send sequence number one greater than the last one received on that logical channel. The receive sequence number, provided by the subscriber on each packet, can thus be used for acknowledging, at this level, the acceptance of packets up to the stated number, in a manner wholly analogous to the usage of the corresponding fields in the link level protocol. Also in an analogous manner, the "RR" packet format allows the subscriber machine to tender an explicit (unpiggy-backed) packet level acknowledgement.

Further, the network can protect itself against flooding by any particular subscriber, by restricting subscribers to a fixed "packet level window size" w, at subscription time. The user machine then agrees not to send more than w packets in, on any logical channel, until an acknowledgement (in the form an an incremented receive sequence number) is received from the destination user.

We have deliberately avoided here the confusing, though perhaps necessarily formal, terminology of the X25 document, which calls the subscriber machine a Data Terminal Equipment or DTE, and the EndNode, to which the user is connected, a Data Circuit-terminating Equipment or DCE.

Within Vicnet itself this is all implemented, up to our understanding of the finer points. Thus our EndNodes handle call requests and call clear packets on those logical channels assigned to VC service, and data transfer packets on all channels; it recognises the P(S) and R(R) fields on data transfer packets, and the P(R) on RR packets, providing a standard packet level window size of two, which applies at the moment to all subscribers.

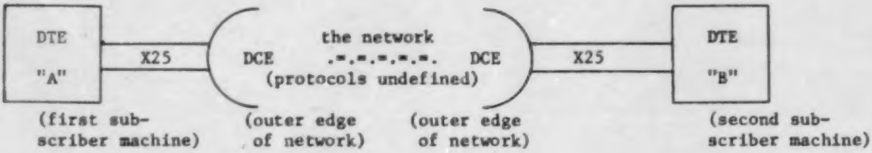
Implementation of the call request required us to define some "telephone numbers" - formally, "DTE addresses" - for our present subscribers. We have opted for using "subscriber names", which are currently CSCB6700, INFOOPO, AMD, MASSEY, B6700, and PO. These are given in the call request packet in the space provided for the "destination DTE address", and the "address length" field is set to twice the number of characters in the name, seeing that the X25 spec recommends the use of digit-oriented addresses. We plan later to move in a "subscriber number" scheme, while retaining the names and providing an automatic "directory enquiry" service for those subscriber machines that haven't learned their correspondents' numbers.

The subscribers themselves have not all implemented all of these features. As it seemed easiest to first implement PVCs, most subscribers are still in transition somewhere between only making use of the PVC capability, and having full call request facilities. The only complete implementation of the protocol in a subscriber currently is in the PDP11 belonging to the Information Science Department.

The hallmark of Vicnet is in fact this orientation to X25. Vicnet, having been designed after October of 1976 when X25 was ratified by the plenary assembly of the CCITT, embraces the standard not only in the letter of its recommendation for a user-network interface, but as a viable suggestion for the internal structure of a packet switching network - the routing, the subscriber name space, the call setup procedure in the communications subnetwork, et cetera. For a nitty gritty example: the X25 specification clearly intends, between the lines, that the "call request" packet from the calling subscriber be allowed to propogate right through the network, all the way to the called subscriber, where it is seen as an "incoming call packet". (The formats for call request and incoming call are, significantly, identical). Yet the spec itself can never say that the call request is literally forwarded through the net from end to end - because, if you think about it, some old networks, with previously established call setup arrangements, might not find it convenient to do it that way. So X25 only delineates, formally, that which transpires between the EndNode and the subscriber machine (between DCE and DTE), while strongly implying or suggesting much more. It is this "much more" that Vicnet takes direct and opportunistic advantage of - we route the call request, for example, right through the source subscriber to destination subscriber,

142

unchanged. Vicnet's genus and differential are truly "an X25 packet switching network".


DTE
"A"
X25
the network
DCE
.=.=.=.=.=.
DCE
(protocols undefined)
X25
DTE
"B"
(first subscriber machine)
(outer edge of network)
(outer edge of network)
(second subscriber machine)

```
call      call request=11 ------->
setup                                -->incoming call=11
phase                                <--call accepted=15
          call connected=15 <-------

          data transfer (out) --->
                                     --> data transfer (in)
          data transfer (out) --->
data                .
transfer            .
phase               .                --> data transfer (in)
                                     <-- data transfer (out)
        data transfer (in) <-------


call    clear request ------------->
clearing                             --> clear indication
phase                                <-- DTE clear confirmation
        DCE clear confirmation <--
```

(a) Call Setup and Clearing for Virtual Calls

first octet	general format identifier =1			logical channel group number
second octet	logical channel number			
third octet	P(R)	m	P(S)	0

Packet Receive Sequence Number
More Data Bit
Packet Send Sequence Number
Zero says "Data Transfer"

(b) Packet Header for Data Transfer Packets

FIGURE 2 X25 PACKET PROTOCOL

A CASE STUDY: THE IMPLEMENTATION OF A TERMINAL INTERFACE PROCESSOR

The major thrust in the early stages of implementation of Vicnet was in the development of subscriber software. There were several reasons for this. Firstly, we did not have the network hardware yet and so were restricted to development on existing hardware. Secondly, the link level HDLC protocol was useful in many non-network applications and the same software could later be used for the entire X25 subscriber interface to Vicnet.

At the time of writing there are five implementations, in varying stages of completion, of subscriber interface software. The Victoria University Computing Services Centre, with some help from Massey University Computer Centre, and the Applied Maths Division of DSIR, have completed implementation of link level software for a B6700 mainframe. Massey are implementing all levels of subscriber software under their "Switcher II" operating system for a PDP11/10. The Applied Mathematics Division of DSIR have a partially completed implementation of link level software for an HP2100 running Time Shared Basic, initially to be used to upgrade a direct connection to the University's B6700. The Information Science Department at Victoria University have complete interface software for a PDP11/10 running the 'Post Office' operating system and are also developing packet level software to complete the interfacing of an M6800 micro computer running the 'Microps' operating system (like the 'Post Office', developed locally).

As is evident from the above, the operating system running on a machine is a primary factor in interfacing that machine to a network. Most machines are not designed for data-communications and will exhibit characteristics which make them difficult to interface. We now offer a case study of a machine whose operating system was designed specifically for networking. Microps(2) was developed for use in a number of 'spots' in Vicnet. Described below is the software required to interface Microps as a Terminal Interface Processor (terminal concentrator) on Vicnet. Microps runs on an M6800 micro computer which, in this case, has 20k bytes of memory and can handle 6 teletypes in addition to its network interface.

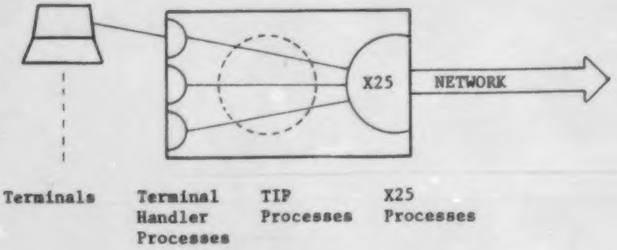


FIGURE 3: OVERALL STRUCTURE OF TIP

Microps allows any number of independant asynchronous processes connected arbitrarily by message queues. X25 is a well-structured 3 level discipline and we have found it advantageous to implement it by using separate processes for each level.

The group of processes that handle X25 level 1 and 2 are a closely integrated unit which can be used in any Microps configuration connected to a HDLC or full X25 line

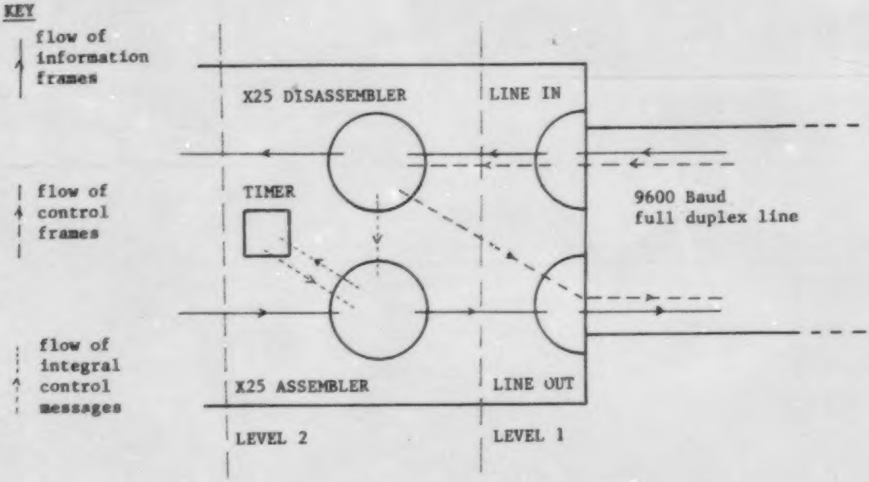

KEY
flow of information frames
flow of control frames
flow of integral control messages
X25 DISASSEMBLER
LINE IN
TIMER
9600 Baud full duplex line
X25 ASSEMBLER
LINE OUT
LEVEL 2
LEVEL 1

FIGURE 4: LINK LEVEL PROCESS CONFIGURATION

Let us follow the path of a typical information frame through the link level process. All frames are bounded by the flag character so the linein process idles until a flag is received. It then stores subsequent characters in a buffer, performing the cyclic redundancy calculation on each one, until another flag is received. Any 'DLE' character is discarded as the transparency indicator described above; the next character is accepted, even if it is a flag or 'DLE', as a data character. If the frame was good (the CRC calculation for the whole frame gave the expected value) the buffer is passed to the next level, the X25DISASSEMBLER process; if the frame was bad then the buffer is discarded and no other action taken.

The X25DISASSEMBLER checks that the sequence number of the frame is what it expects. If so, then the frame header is stripped off, the buffer is passed out of the link processes and a 'RR' (Receiver ready) or "Acknowledgement" frame is sent to the lineout process for transmission. If the sequence number is wrong, then a 'REJ' (Reject) frame is sent and the buffer discarded.

After the buffer has left the link processes the details of level 1 and 2 are no longer relevant; the link processes may be treated as a logical part which provides error-free transmission of messages to and from a distant location (Figure 5).

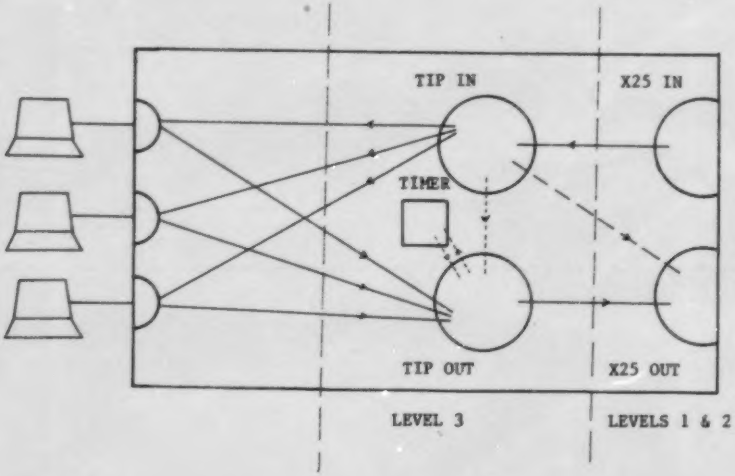


FIGURE 5: X25 LEVEL 3 IN THE TIP

The Terminal Interface Processor (TIP) allows users on terminals to connect into the network. The two processes TIPOUT and TIPIN implement the packet level of X25 as well as providing prompts, diagnostics and control functions for the user.

The unit of activity on an X25 line is the 'call'. A data structure of call records is shared by TIPIN and TIPOUT. A single call record contains the X25 channel of the call, the terminal that made the call, the status of the call (ready, waiting, data transfer) and the packet send and receive variables.

When the user switches on a terminal the TIP prompts the terminal user with 'VICNET TIP, CALL PLEASE'. The user types in a "telephone number", TIPOUT picks a free channel and then sends a 'call request' packet out into the network. The call status is marked as 'waiting for call'. In the normal sequence of events a 'call accepted' packet will be received by TIPIN who will find the relevant call record, mark the call as being in 'data transfer' status and then sends "CALL ACCEPTED, GO AHEAD PLEASE" to the terminal.

All subsequent input from the terminal will be made into a data transfer packet and forwarded. Data packets from the network on that channel will be directed to the terminal by TIPIN.

When the user wishes to terminate the call he enters 'BYE'. A 'clear request' is sent by TIPOUT and the call put into 'waiting for clear' status until a 'call accepted' packet is received by TIPIN. The call is removed and the call record is marked as free.

REFERENCES:

(1) CCITT, "Recommendation X.25 for the Interface Between Data Terminal Equipment and Data Circuit Terminating Equipment", October 1976.

(2) McKendry, M.S. "The Infrastructure of the Microps Operating System", Applied Mathematics Division, D.S.I.R., Technical Report 73, October 1977.

(3) Bowles, S.W. "Vicnet, an X.25 Packet Switching Network", Proceedings of this Conference.

COMMUNICATIONS ASPECTS OF AN AIRLINE RESERVATION SYSTEM

AUTHORS : I.S. Campbell, Communications Technical Controller,
Air New Zealand Ltd, Auckland

L.M.R. Paver, Advisory Systems Engineer,
IBM (N.Z.) Ltd, Auckland

ABSTRACT : This paper discusses communications aspects of Air New Zealand's international passenger reservations, departure control, and message-switching system. Both authors were involved in the installing of the system over a period of nearly three years.

The network has a total of about 400 terminals, mostly visual display stations, in the U.K., Australia, U.S.A., and the Far East. Of particular interest are the message-switching application and the SLC link control. The system operates 24 hours a day, 7 days a week, and is housed at Air New Zealand's new Computer Centre in Newton, Auckland.

C.R. Categories:

3.80, 3.81, 3.83, 3.89

CONTENTS

ENVIRONMENT	1.0
COMMUNICATIONS FUNCTIONS	2.0
EQUIPMENT	3.0
CENTRAL COMPUTER FUNCTIONS	4.0
MESSAGE SWITCHING	5.0
SYNCHRONOUS LINK	6.0
INSTALLATION AND TESTING	7.0
FUTURE DEVELOPMENTS	8.0
REFERENCES	9.0

ACKNOWLEDGEMENTS

Air New Zealand - for permission to present the paper

IBM - for word processing services

Mr R. Penman - for permission to reproduce his cartoon

Mr J. Goodman - for permission to reproduce his drawing

Our colleagues in the IPARS Project - from whom and with whom we learned and shared knowledge and ideas.

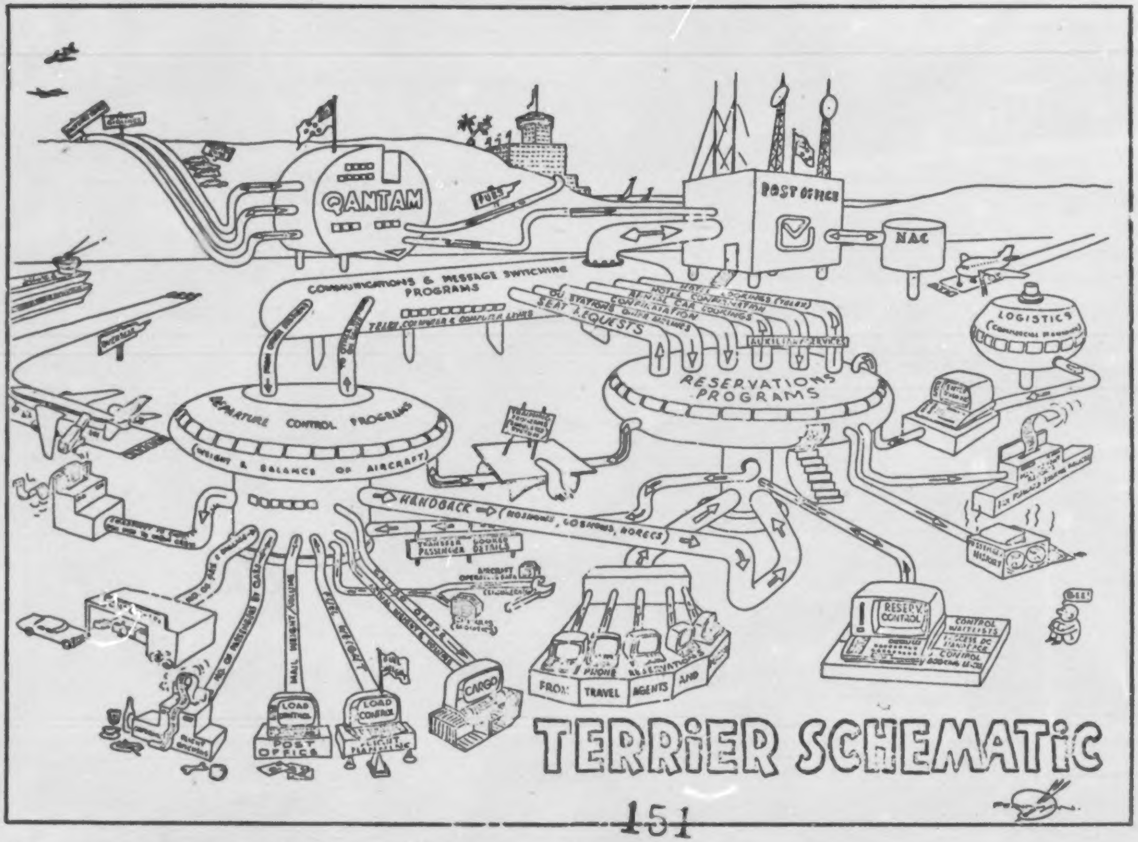
1.0 ENVIRONMENT

The special requirements of an airline's reservation system are for very fast response times to a heavy message arrival rate, for support of industry-specialised terminals, and for very high system availability. Air New Zealand's international reservations system (CARINA-II) and domestic reservations system (CARINA-I) each processes about 7 transactions per second during busy periods. Most messages originate from "agents" at CRT terminals making bookings for passengers who either telephone or call at a booking office. Additionally, international passengers departing from Auckland are checked in at the airport by other agents operating similar sets.

The name of the programming package used is IPARS. This package, or one of a number of its variants, is in use by the majority of the world's large airlines. The control program, ACP (Airlines Control Program) (Ref.1), interfaces with reservations and departure-control application programs. It includes CCP (Communications Control Program) which performs all input and output on telecommunications lines. In Air New Zealand the package is implemented on an IBM System/370 Model 145, with 512K bytes of main storage.

The overriding criteria in design of the IPARS package was that even under peak conditions 90% of the transactions should experience a response time of 3 seconds or less. This criteria has been achieved by providing only those functions which were necessary for supporting reservations and similar applications. For instance, in the communications area, only a limited number of line types is supported, and these are mostly specialised airline types. On the disk storage side, all records have one of only 3 record lengths, and only one access technique is available.

Prior to its merger with National Airways Corporation on 1 April 1978, Air New Zealand's IPARS system was called TERRIER. It had been in planning from early 1975, and was commissioned in stages between November 1977 and February 1978.


QANTAM
FUEL
POST OFFICE
NAC
COMMUNICATIONS & MESSAGE SWITCHING PROGRAMS
TELEX, COMPUTER & COMPUTER LINKS
HOTEL BOOKINGS (TELEX)
HOTEL CONFIRMATION
RENTAL CAR BOOKINGS
CONFIRMATION
OUTSTATIONS QUERIES
SEAT REQUESTS
AUXILIARY SERVICES
RESERVATIONS PROGRAMS
LOGISTICS
(Commercial Planning)
DEPARTURE CONTROL PROGRAMS
(WEIGHT & BALANCE OF AIRCRAFT)
HANDBACK
(NOSHOWS, GOSHOWS, NORECS)
TRANSFER PASSENGER
LOUNGE DETAILS
AIRCRAFT OPERATING DATA
ENGINEERING
NO OF PASSENGERS BY CLASS
MAIL WEIGHT/VOLUME
FUEL WEIGHT
CARGO
LOAD CONTROL
POST OFFICE
FLIGHT PLANNING
PHONE RESERVATIONS AND
FROM TRAVEL AGENTS
RESERV. CONTROL
CONTROL WAITLISTS
PROCESS OF STANDBY
CONTROL BOOKING LIMITS
PASSENGER HISTORY
GEE!

TERRIER SCHEMATIC

During the planning phase the project head-count averaged 20 programmers, 8 analysts seconded from user departments, and 3 administrators. In particular, the communications/message-switching area was supported by 1 full-time and 1 part-time person from the Communications department, 2 programmers to handle CCP, 2 programmers to handle Message-Switching, with support to both areas from an IBM Systems Engineer.

Not a great deal of program coding was done, but many changes and corrections were made to the distributed code before implementation, and much time was required for setting up tables and generally tailoring the package to Air New Zealand's needs. Extensive program testing was done, and fortunately a good selection of IPARS test tools were available for help in this area. Testing of communications hardware was much simpler than it might have been, since most terminals, lines and concentrators had already been in use by the company when connected to the QANTAM reservations system in Sydney, where the Air New Zealand database was co-resident with that of the Qantas Airline.

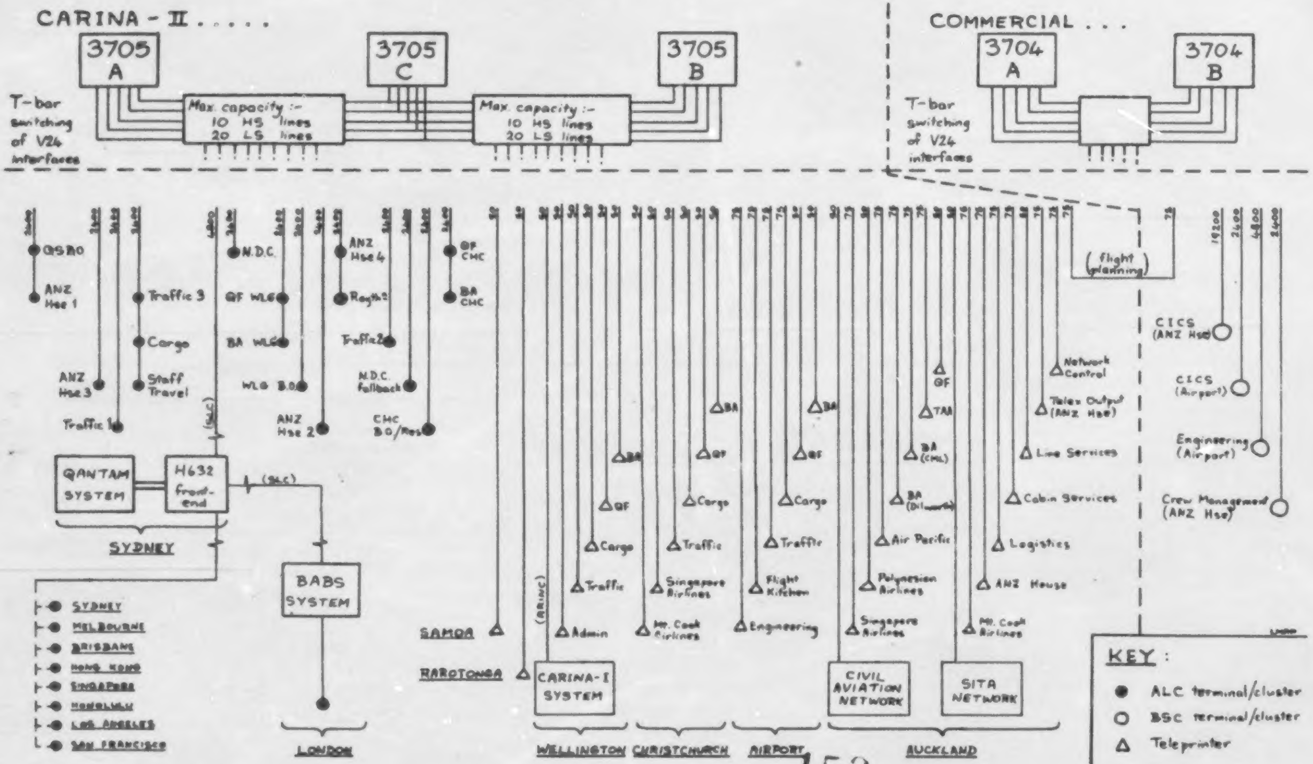
2.0 COMMUNICATIONS FUNCTIONS

There are four types of line control used by Air New Zealand but only two basic types of terminal, namely, agent set and teleprinter. The line control types not directly involving terminals are the link protocols for a high-speed link to the QANTAM computer and a low-speed link to the Wellington CARINA-I computer.

2.1 ALC

Airlines Line Control (ALC) drives agent sets (display stations) and their associated buffered printers. This is a special protocol first developed for the IBM 1006 agent set on the American Airlines SABRE system commissioned in 1963 and subsequently implemented on other terminals (Ref.2). ALC uses true full-duplex communication, that is to say, one terminal

COMMUNICATIONS NETWORK — INTERNATIONAL



CARINA - II . . . . .
3705 A
3705 C
3705 B
T-bar switching of V24 interfaces
Max. capacity :- 10 HS lines 20 LS lines
Max. capacity :- 10 HS lines 20 LS lines
COMMERCIAL . . .
3704 A
3704 B
T-bar switching of V24 interfaces
QSRO
ANZ Hse 1
Traffic 3
Cargo
Staff Travel
ANZ Hse 3
Traffic 1
N.D.C.
QF WLG
BA WLG
WLG B.O.
ANZ Hse 4
Reg'n
Traffic 2
N.D.C. fallback
ANZ Hse 2
QF CHC
BA CHC
CHC B.O./Res
(SLC)
QANTAM SYSTEM
H632 front-end
SYDNEY
BABS SYSTEM
LONDON
SYDNEY
MELBOURNE
BRISBANE
HONG KONG
SINGAPORE
HONOLULU
LOS ANGELES
SAN FRANCISCO
SAMOA
RAROTONGA
(ARINC)
CARINA-I SYSTEM
Admin
Traffic
Cargo
QF
BA
Mt. Cook Airlines
Singapore Airlines
Traffic
Cargo
QF
BA
Engineering
Flight Kitchen
Traffic
Cargo
QF
BA
Singapore Airlines
Polynesian Airlines
Air Pacific
BA (Dilworth)
BA (CML)
TAA
QF
Mt. Cook Airlines
ANZ House
Logistics
Cabin Services
Line Services
Telex Output (ANZ Hse)
Network Control
(flight planning)
CIVIL AVIATION NETWORK
SITA NETWORK
WELLINGTON
CHRISTCHURCH
AIRPORT
AUCKLAND
19200
2400
4800
2400
CICS (ANZ Hse)
CICS (Airport)
Engineering (Airport)
Crew Management (ANZ Hse)
KEY :
● ALC terminal/cluster
○ BSC terminal/cluster
△ Teleprinter

153

may be transmitting its buffer in to the CPU while another terminal in the same line is being polled or is receiving an output message. Each message consists of a 2-character synchronisation pattern, a data area, and (usually, a cyclic check character. In order to make most efficient use of transmission time, each character consists of only 6 bits, giving a 64-character set.

ALC terminals are referred to as "high-speed", but the term is really a misnomer, as the speed is only 2400 bps, (or 400 characters per second). The philosophy of the ALC network is based on several key factors :

1. For every input to the CPU there will be a response to the agent set;

2. If an agent makes an entry to the system and does not receive a reply (because of system or communications failure), he will try again after several seconds, and therefore the system does not need to send ACK and NAK signals to the terminal;

3. If the reply message is garbled so that the terminal detects a transmission error indicated by a warning light, the agent can be expected to request a retransmission from the CPU and therefore the terminal does not need to send regular ACK/NAK signals to the CPU.

The above factors eliminate the rigorous transmission checking which tends to slow down the performance of other line protocols.

Most ALC terminals are used by reservations or departure-control agents. Reservations agents operate either across-the-counter sales in a booking office or sales to customers who telephone a reservations office. Departure-control agents are the check-in clerks at the airport, and the behind-the-scenes airport staff responsible for weight-and-balance checking of all departing aircraft. However, a number of special users all requiring terminals are worthy of mention. Both reservations and departure-control systems need controllers whose job is to administer their respective agents and to perform major management functions on their database - such as entering base information about new flights, or transferring a particular flight from the reservations stage to the departure-control stage. Slightly different are the controlling operations performed by the Network Control Centre (NCC) for Message Switching and communications in general, and by the Computer Room operators (CRAS) for ACP control and on-line debugging.

Agent sets are clustered usually about 10 or 15 per concentrator. Often there are a few buffered printers attached to these concentrators, to allow agents to obtain hard-copy of information currently displayed at their screen, or specially extracted from the database. In some cases they may also be designated to receive messages from the (low-speed) Message-Switching system.

2.2 TTY

The other type of terminal is the teleprinter. Its use is quite different. There are about 40 teleprinters, a number of which are in the offices of other organisations associated with the Airline. Teleprinters (referred to as TTY) are usually unattended, so output is always on hard-copy (and occasionally on paper tape). It is quite possible for several messages to have been received by a TTY, and to be waiting for distribution. One teleprinter may deliver messages for several different people or departments, so consequently each message must contain an address. A 7-character address is standard. The address may be simply that of the terminal, or it may indicate one of the users served by the terminal. Similarly, each message must include the address of the originator and the time of despatch.

Teleprinters use start/stop transmission, with International Telegraph Alphabet No 2, (to which Baudot code is fairly similar). This code contains only 5 bits. However, there are figures-shift and letters-shift characters which instruct the receiving machine how to interpret the succeeding characters, thus nearly doubling the character set which may be transmitted. None of the 5 bits is used for parity, and there is no cyclic check character. As a result, there is no way the hardware of the receiving machine can detect a transmission error. It becomes the responsibility of the recipient of any garbled message to ask for retransmission by the system, or even to send a message to the originator asking for clarification. These requests are handled by the Message Switching package (Ref.3). Compared to ALC messages, TTY messages receive much more careful handling. The system inserts a sequence number in every output message on a given line, and it expects all input messages on a given line to follow an unbroken sequence. Any discrepancy causes an error to be notified to the NCC controller who monitors the low speed communications network. A TTY operator noticing any discrepancy will send a message to NCC asking for retransmission of any missing messages.

The format of TTY messages is strictly laid down in an international standard by IATA (International Air Transport Association) (Ref.4) and every time a TTY keyboard is operated, this standard must be adhered to. Briefly, each message consists of one or more addresses, an origin, date and time, circuit sequence number, and text. Many TTY messages are generated not by a keyboard operator, but by a remote IPARS system (say QANTAM) in which one of its agents tries to book an Air New Zealand flight, and so causes that system to generate and despatch a TTY message to Air New Zealand. The text part of these inter-airline reservations messages is subject to a further international standard, AIRIMP (Ref.5).

TTY transmission is either full-duplex (input and output concurrently and independently), half-duplex (one way at a time), or simplex (output only). Some circuits operate at 75 bps, and the remainder at 50 bps. Full-duplex operation is achieved by connecting to one line a printer-only terminal (for output) and a keyboard-printer (for input). Half-duplex operation involves the operator and the CPU initially contending for the line. In all cases, transmission continues from one message to the next without needing any response from the receiving end, a mode of operation described as "freewheeling".

2.3 SLC

This was previously referred to as the line-control without a terminal. In fact, it is only ever used for point-to-point transmission between two CPU's. SLC (Synchronous Link Control) protocol is defined by IATA (Ref.4), for use in "store and forward" message transmission between two CPU's, over a link consisting of from 1 to 7 parallel voice-grade channels. It uses an elaborate program-controlled procedure, with full-duplex data flow, and complete data integrity through automatic error-detection and retransmission. The character set is ASCII (7 bits plus parity).

Air New Zealand uses SLC on its trans-Tasman link to the QANTAM computer. This is a single-channel, full-duplex, 4800 bps circuit, at present using half the bandwidth of a 9600 bps satellite link between the two countries. The SLC protocol caters for two types of message, each having different transit-time requirements. The less urgent (Type B) is the inter-airline TTY message already mentioned. The more urgent (Type A) is the message to or from an overseas agent set or printer. There is no harm in delaying a TTY message for some tens of minutes if necessary, but an agent needs a response of the order of a few seconds. Consequently Type A messages are always queued ahead of Type B messages.

The theory and implementation of the SLC link are sufficiently interesting to warrant a fuller discussion, which appears in Section 6.

2.4 ARINC

This form of link control is more correctly defined as asynchronous processor-to-processor link control. The term ARINC is derived from the name of a large message-switching utility company operating within the United States, namely Aeronautical Radio Inc. Air New Zealand employes this type of link on its TTY line to the CARINA-I installation (ex-NAC) in Wellington. Electrically, the circuit is indistinguishable from a full-duplex TTY circuit. It operates at 50 bps (as do all the CARINA-I TTY lines), and its messages have almost the same format as those sent or received from terminals. Traffic in both directions is sequence-checked by the system, and any out-of-sequence conditions notified to the NCC controller. But its distinguishing feature is the use of simple control blocks, as follows.

The VA (Validate Action) control block, comprising 8 fixed characters, is transmitted at intervals of about 20 seconds when no traffic is waiting for output. This provides assurance to the remote centre that the system is operationally ready to accept input messages. The VA is also transmitted after each message.

One centre may wish to warn the other to stop sending, perhaps because messages are arriving in faulty condition, or perhaps because no messages or control blocks have been received for about a minute. In this case an IA (Invalidate Action) control block is sent at 20 second intervals.

3.0 EQUIPMENT

The following are the major items of equipment. Every piece of equipment is backed up by a standby unit, sometimes on a one-for-one basis, sometimes on a 2-out-of-3 basis, etc.

3.1 CPU

Heart of the CARINA-II system is a System/370 model 145 CPU. This model was chosen, as having sufficient instruction execution speed to handle any likely peak load within the next several years. Its 512K storage is large enough to contain the most frequently accessed programs and data, thus avoiding excessive use of the disk access channel. There are three I/O channels supporting disks, tapes and communications, respectively. In the event of CPU failure, an identical backup CPU could be switched in, in a matter of seconds, to take its place. This backup CPU normally performs offline IPARS support and program development work using CMS (Conversational Monitor System), and also is available if needed as backup to the System/370 model 158 used for batch data processing.

3.2 DISKS

The system uses 10 drives of IBM 3340 disk storage whose removable modules include not only disks but also the read/write heads for those disks. The 3340 was chosen on account of the extremely high reliability resulting from this sealed head arrangement. Each module has 696 "cylinders", of which 5 have fixed heads. The most commonly used datasets have been placed under the fixed heads and in the middle range of cylinders, resulting in an average seek time considerably better than the manufacturer's rating.

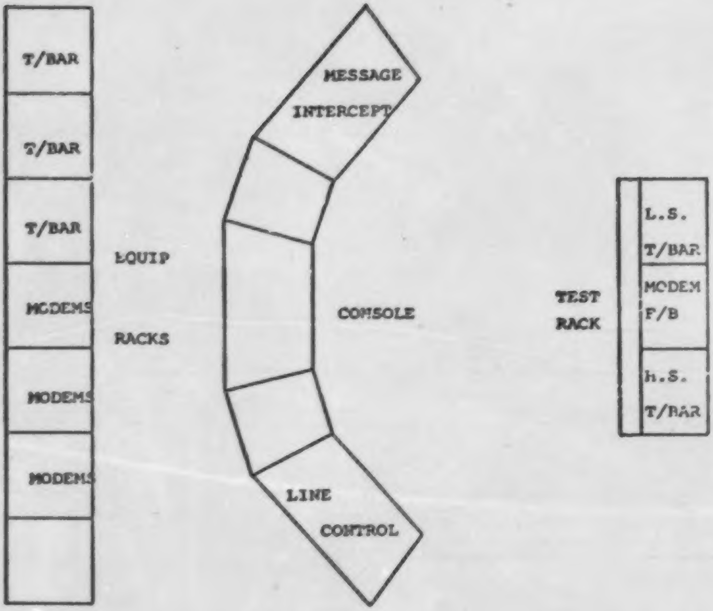
3.3 TAPES

A bank of seven 3420/3 tape drives is available for logging and various utility functions.

3.4 COMMUNICATIONS CONTROL

The path between computer and transmission equipment is through 3705 controllers and the standard V24 interface. Among the 3705 functions are data serialisation, computing cyclic check

CARINA 11 NETWORK CONTROL AREA.


T/BAR
T/BAR
T/BAR
MODEMS
MODEMS
MODEMS
EQUIP
RACKS
MESSAGE
INTERCEPT
CONSOLE
LINE
CONTROL
TEST
RACK
L.S.
T/BAR
MODEM
F/B
H.S.
T/BAR

characters, scanning for embedded control characters, and clocking low-speed lines. Its functions are performed by a program (the "emulator") which runs in 16K of storage, and can be refreshed, if necessary, from the ACP system. The emulator program supports control panel diagnostics of both internal and simple down-line operation.

3.5 NETWORK DESIGN

As Air New Zealand's reservations and message switching was already on-line and multi-hosted in QANTAM, the basic network was already in existance. However, the Qantas philosophy of front end processing and nodal concentration differed from that required for the new TERRIER system. The nodal concept was retained, with the major nodes in Wellington and Christchurch. All circuits in the respective spheres of operation of each nodal point were terminated at the node and from there connected back to Auckland by multi-stream 9600bps modems.

The low speed circuits were multiplexed on to a single 2400 bps channel of the modem, leaving three channels for data transmission. Roll-call polling has been used to multi-drop Terminal Interchanges on a single line thus giving more capacity.

3.6 MODEMS

The modems used on the network were basically dictated by those used on the QANTAM network and are mainly the Racal series. Main trunk circuits use the Racal 96MM. This is a multi-stream modem operating at 9600 bps, as either a single channel or up to four 2400bps channels. In the CARINA-II network it is used in a four-channel point-to-point full-duplex mode connecting the nodal points back to Auckland. For the terminal sites the Racal 2400bps modems are used. In the central site Racal CS24 modems are used. These are 2400bps card modems which are mounted eight to a rack and have a common power supply and monitoring facilities. In the field are used the 24LSI and the old 2200/24 modems. All these modems are compatible and can be interchanged with ease.

On the Auckland/Sydney circuit a Codex 9600C multi-channel modem is used. This is again dictated by QANTAM requirements. This modem is configured to provide one channel at 4800 bps for the SLC link and two channels at 2400bps, one for servicing the Qantas and British Airways terminals in Auckland whose terminal equipment is incompatible with CARINA-II, and the other as a test channel for ongoing development between the Airlines.

Provision has been made with standby modems in the prime locations to keep down time to a minimum in the event of modem failure.

3.7 SIGNALLING CONVERTERS

The V24 to 80-0-80 conversion for asynchronous data used on the teleprinter circuits is achieved by use of the Racal DA7330A Signalling Converter. This consists of a card frame which houses a power unit and twelve converter boards. Line inputs and outputs are directed through a switching strip which allows line currents to be directly monitored, and for each channel a potentiometer in series with one wire enables the receive line current to be adjusted. Corresponding to each card are four terminals on the switching strip. These allow for independent connection per channel of two-wire line, four-wire line, two-wire line with earth return, or single-wire line with earth return. Each card can be configured to operate full-duplex or half-duplex, and at any speed required.

3.8 TERMINAL SYSTEM

The terminal system used is the Raytheon PTS100 series. This system is a central processing unit which can drive from 1 to 48 display screens, depending upon the system used. The processor is a general purpose 16-bit mini-computer and the display screens can be driven either direct from the processor or from a remote concentrator. The remote concentrator is designed to drive displays located up to 4000 feet from the processor.

The displays used have a screen capacity of 15 lines each 64 characters long, giving a total of 960 characters. The keyboard is a standard 67-key model. An indicator panel is included which has an eight unit status giving the state of the system for operator information.

Two different types of printer are in use, the type located in any area being determined by the particular requirements. The main printer is the Extel 30cps matrix printer. For high load areas the Centronix 100cps printer is used. Neither printer has a keyboard. Information is transferred to the printer either by a local print facility in the PTS100 or by transmission from the ACP System.

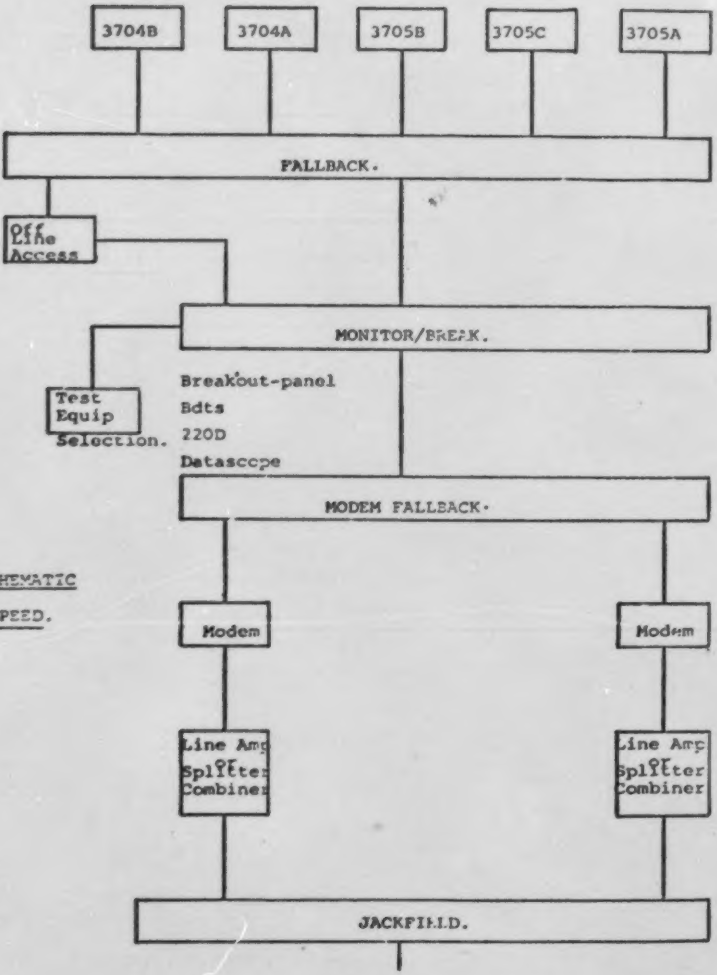
3.9 SWITCHING

The switching system in the Network Control Centre provides operating personnel with the means of reconfiguration of the data communication channels for backup V24 switching to the 3705's, for interface to the testing equipment, and for backup switching to spare channels and modems. This switching is achieved by use of T-Bar switches, whilst the line switching and monitoring is achieved by conventional jackfield techniques.

The T-Bar switching has been designed to provide the following functions :

a) Switch two groups of 16 on-line modems to any two out of three 3705 transmission control units.

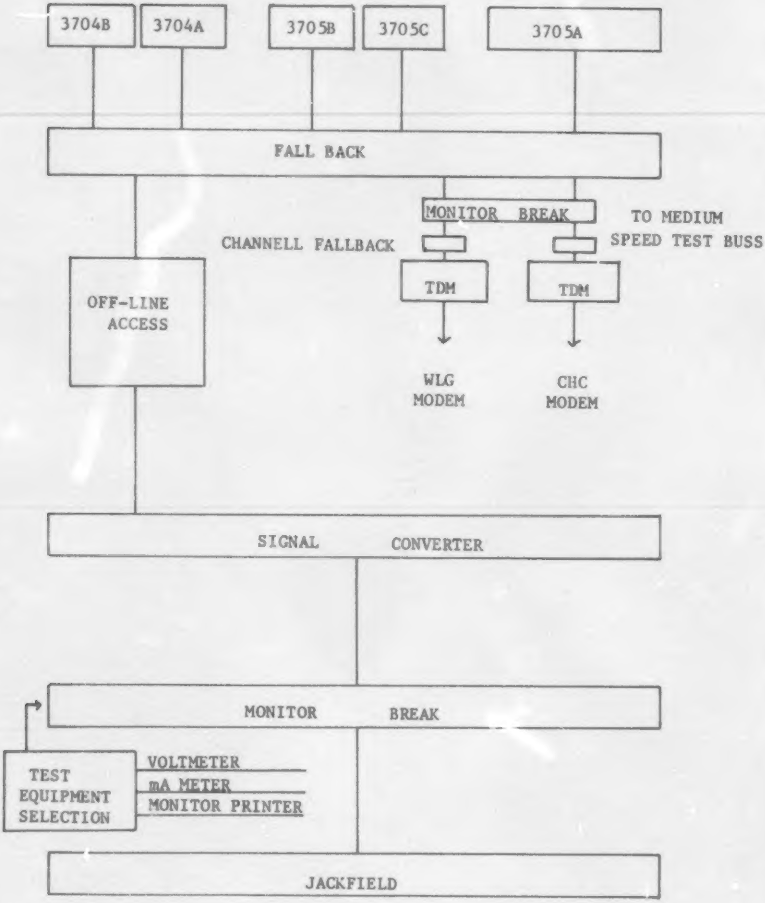
b) Bypass one of 32 on-line modems from its on-line TCU to a DTE emulator for testing.


3704B
3704A
3705B
3705C
3705A
FALLBACK.
Off Line Access
MONITOR/BREAK.
Test Equip Selection.
Breakout-panel
Bdts
220D
Datascope
MODEM FALLBACK.
Modem
Modem
Line Amp or Splitter Combiner
Line Amp or Splitter Combiner
JACKFIELD.

BLOCK SCHEMATIC

MEDIUM SPEED.

LOW SPEED - BLOCK SCHEMATIC.


3704B
3704A
3705B
3705C
3705A
FALL BACK
MONITOR BREAK
TO MEDIUM SPEED TEST BUSS
CHANNELL FALLBACK
OFF-LINE ACCESS
TDM
TDM
WLG MODEM
CHC MODEM
SIGNAL CONVERTER
MONITOR BREAK
TEST EQUIPMENT SELECTION
VOLTMETER
mA METER
MONITOR PRINTER
JACKFIELD

c) Simultaneous monitoring of 24 EIA signals of the bypassed modem.

d) Connect the "test" modem to any one of 16 ports of the TCU currently in the "spare" status.

e) Monitor on a continuous basis, three discreet EIA interface signals of each on-line modem via light emitting diodes (LED).

f) Connect a bypassed on-line modem to one of 16 ports of the spare TCU.

g) Bypass any on-line modem and connect the circuit to the standby modem of the same type.

Further switching has been provided to enable the audio monitoring of any circuit and to fallback splitter/combiners and line amplifiers.

3.10 TESTING TOOLS

A wide range of testing equipment has been provided to maintain the communications network to a high degree of efficiency. Besides the normal monitoring and metering facilities are included the Datascope and Buffered Data Transmission Simulator (BDTS). These two items of equipment provide a very powerful tool for error isolation, the Datascope for monitoring the line and the BDTS for simulating either a Terminal or the CPU.

A terminal interchange (TI) and a number of terminals are dedicated to the Model Airline System. This is used to test all improvements and changes to the message switching and communications programs.

4.0 CENTRAL COMPUTER FUNCTIONS

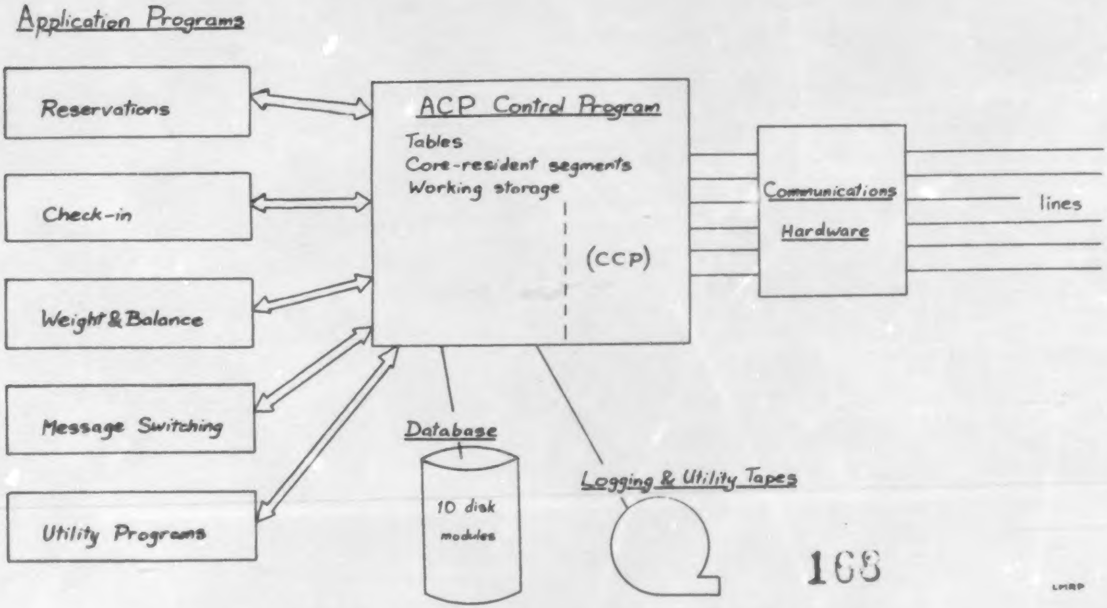
All communications functions are performed by CCP (Communications Control Program) which is an integral part of ACP. This is different from most other systems, where the control program usually interfaces to a generalised "TP method" which performs communications input/output. Although only a limited number of line types is supported, the integration of CCP results in very efficient control and a relatively short instruction path.

The core-resident part of ACP is about 112K bytes, of which CCP accounts for 33K. All other programs are written in segments of up to 1055 bytes. This includes all application programs (Reservations, Departure Control, Message-Switching), the major part of CCP, and a substantial part of ACP. A small number of these segments (the most frequently used) can be loaded into main-storage at IPL time, but for the vast majority they must be fetched into working storage whenever needed. Segments are always re-entrant, and can therefore be used for several messages concurrently.

The unit of work in the CPU is the "entry" which is triggered by the receipt of a message. Several entries are likely to be active at one time, though the life of each is perhaps only a few seconds, or a fraction of a second. Each entry will proceed until it has to wait (usually) for disk-file reference, at which point another entry will be dispatched. In handling a customer's requirements there may be an average of 5 to 10 messages sent in by the agent during the conversation. Each of these results in a separate entry. However, data which is being built up throughout the conversation is held in a medium-term block of working storage called an AAA (Agent Assembly Area).

The continued receipt of messages is ensured by polling ALC lines, and by keeping all other line types prepared to receive, with input buffers ready.

IPARS STRUCTURE


Application Programs
Reservations
Check-in
Weight & Balance
Message Switching
Utility Programs
ACP Control Program
Tables
Core-resident segments
Working storage
(CCP)
Communications
Hardware
lines
Database
10 disk
modules
Logging & Utility Tapes

When an interrupt on an input line signals receipt of a block of data, CCP checks whether it is part of a multi-block message, in which case it must be chained to the other parts. When the whole message has been received, CCP triggers a program called OPZERO which creates a new "entry" and places it on a queue called the "input list".

Each entry waiting for CPU processing is represented by an item in one of three queues. Top dispatching priority goes to the "ready list" consisting primarily of entries which have just completed an I/O request (disk or tape). Second priority goes to the "input list" mentioned above, since ACP prefers to complete work in progress before beginning new work. Lowest priority is given to the "deferred list" where entries are placed when the application program decides there is no particular time constraint, or when there is to be a wait for some time-initiated event. All these lists are processed with interrupts unmasked. There is no mechanism for pre-empting an entry, once dispatched, while it is still using CPU cycles. But to protect against looping, any entry using as much as 500 milliseconds of CPU time is automtically terminated.

The primary communications support in ACP is for ALC terminals. This factor has had a dominant influence in the architecture of ACP. Polling of ALC lines is triggered on a regular time-initiated basis. It is possible to assign different intervals to different lines, but in CARINA-II an interval of 200ms for all lines has been found satisfactory. If ever a particularly heavy load should cause the pool of working storage to drop to a threshold level, polling would be inhibited until the condition was alleviated, and it became safe to solicit more input messages. Polling routines also cater for terminals which fail to respond to a poll sequence. Rather than stopping polling altogether, they continue to poll the troublesome terminals at a reduced rate until the problem is fixed.

CCP service is requested by an application program issuing an SVC (Supervisor call) instruction. Most commonly this is done to request transmission of an output message. The application program identifies the message, its format, and the line number to be used. At this point the message joins an output queue for the line, and the application program loses control of it. It never needs to know when the actual transmission takes places. CCP handles each I/O request in 4 phases - "Initiate" phase which builds CCW's and issues the SIO; "Interrupt" phase which checks status at completion of I/O; an optional "Post-Interrupt" phase which performs low-priority housekeeping associated with completion; and an "Error" phase to be used when needed. For each line a branch vector indicates which phase is currently active on the line.

Communication lines are addressed by their symbolic line numbers (SLN). There is a unique two-digit number for each line, with various ranges of numbers denoting the various line types. The SLN is used both internally, by an applications program requesting service, and externally by the computer operator performing a control function on the line. In order to allow an ALC terminal to be physically swapped over to a different line without requiring table-changes in application programs, a single CCP table gives a mapping of the internal to external SLN's and terminal poll addresses.

One of the key characteristics needed in a high-availability system is a rapid restart procedure for use following system failure. A restart time of around 20 seconds is achieved, largely due to the availability of "keypoint" records. Under this technique, a large amount of important system information (about 10K bytes) is written to disk every 5 minutes. This information is arranged to be kept in adjacent main storage fields, from which it may easily be written to disk or, when restart is needed, read back in. There are other restart functions of course, such as clearing up the work which was in progress when failure occurred, and re-starting those lines which were previously started.

It is vital that the database can be replaced in the event of hardware malfunction on a disk drive, or even possible corruption by software. To this end, Air New Zealand captures the contents of every IPARS disk once every day. This data is read from disk by a program known as Capture, and is written to backup tape. The operation is performed by a utility program running under ACP, so that there is no disruption to IPARS availability. In order to re-establish the database it is necessary to restore the backup tape to the corrupted disk, then reprocess all the present day's logging tapes, so that recent transactions may be re-applied to the files.

5.0 MESSAGE SWITCHING

A passenger requesting a flight on an airline will normally require a return booking, hotel accommodation, or even a rental car at his destination or at intermediate stops. The airline industry also has many occasions on which messages have to be passed to other airlines, to airline operating agencies, and between departments of the airline itself.

To enable distribution of these messages several communication networks have been established. These networks are normally run on a co-operative basis and cover different areas of the globe. To ensure a world-wide coverage, an airline must become a member of the co-operative in its sphere of operation. Air New Zealand belongs to the major networks as follows :

a) OFTS (Overseas Fixed Telecommunications System)
This network is a world-wide system set up for the use of a group of 22 airlines based in Europe, Africa, Asia, America and Australasia.

b) SITA (Societe Internationale de Telecommunications Aeronautique)
An association of some 130 airlines mainly outside the United States, providing a network covering most areas of the globe but with its main concentration in Europe.

c) ARINC (Aeronautical Radio Incorporated)
Provides an interline message switching function in the United States.

5.1 FORMAT

The above networks require messages to be prepared in accordance with a common format. A message consists of the following parts :

Heading Section : which consists of a starting pattern, start-of-message signal, the circuit identifier and the input sequence number. This information is added to check the sequence of the incoming message and effect retrieval from the transmitting station.

Address Section : consists of the start-of-address signal, priority indicator, and up to 32 addresses followed by the end of address signal. The priority indicator determines the order in which the message is to be handled. The address consists of seven alpha characters. These characters represent :

- Three characters for the City of Destination
- Two characters for the Office within that City
- Two characters for the Airline

Origin Section : contains the seven alpha character indicator of the office of origin, using the same construction as that of the address. It may also be followed by other information required by the originating office provided that the section does not exceed 69 characters.

Text Section : is composed entirely by the originator of the message and is limited only by length and the character sequences which correspond to the message control sequences. The text section is terminated by the end-of-text sequence.

Ending Section : The system will accept and generate one of a number of ending sections depending upon the options set for the particular line.

```
ZCZC ATA123<≡↓                      Heading Section
QK SYDRMQF<≡↑                       Address
.AKLRMTE<≡                          Origin
TEXT OF MESSAGE<≡                   Text
=<≡
   ≡                                End-of-Text sequence
   ≡
NNNN                                Ending
```

The message switching package has been designed to accept messages and check them for the above format. Those messages which pass this check are then routed in accordance with the pre-determined routing tables. Those which do not pass the check are delivered to an intercept queue for manual processing where they are modified to the acceptable standard or are rejected back to the originator for re-entry.

5.2 ROUTING

Each major switching network has fixed routing patterns to enable messages to be transmitted through the network in the most economical way. Therefore, for each message received, responsibility for routing may exist for one or more addresses within the message. Each message is checked to see whether there is routing responsibility for the address, given the originator and input line. If routing responsibility does not exist the address is ignored. If any of the addresses and the originator constitute third party traffic then the whole message will be rejected.

Provision also exists, due to circuit and centre outages, to bypass the pre-determined routing. This is achieved by two separate functions :

Alternate Routing, when the messages require to be sent from the system to a circuit other than that required for normal routing.

Diversion Routing, when messages are received by the system on circuits other than those on which they would normally be received.

5.3 REPORTS

The reports generated by the system are mainly statistical, and can be printed out at preset time intervals and/or in response to a command.

Status and Traffic Report contains the cumulative totals of messages sent and received by each line/terminal for the current day and also the current status of each line/terminal.

Line Status Report gives the sequence number of the last message transmitted and the last received on the line/terminal quoted in the command.

Line Utilization Report contains the daily cumulative count of messages sent and received for each line.

Queue Report includes all the low speed lines in the system and indicates the number of messages in queue for the lines. This report can additionally be called for individual lines.

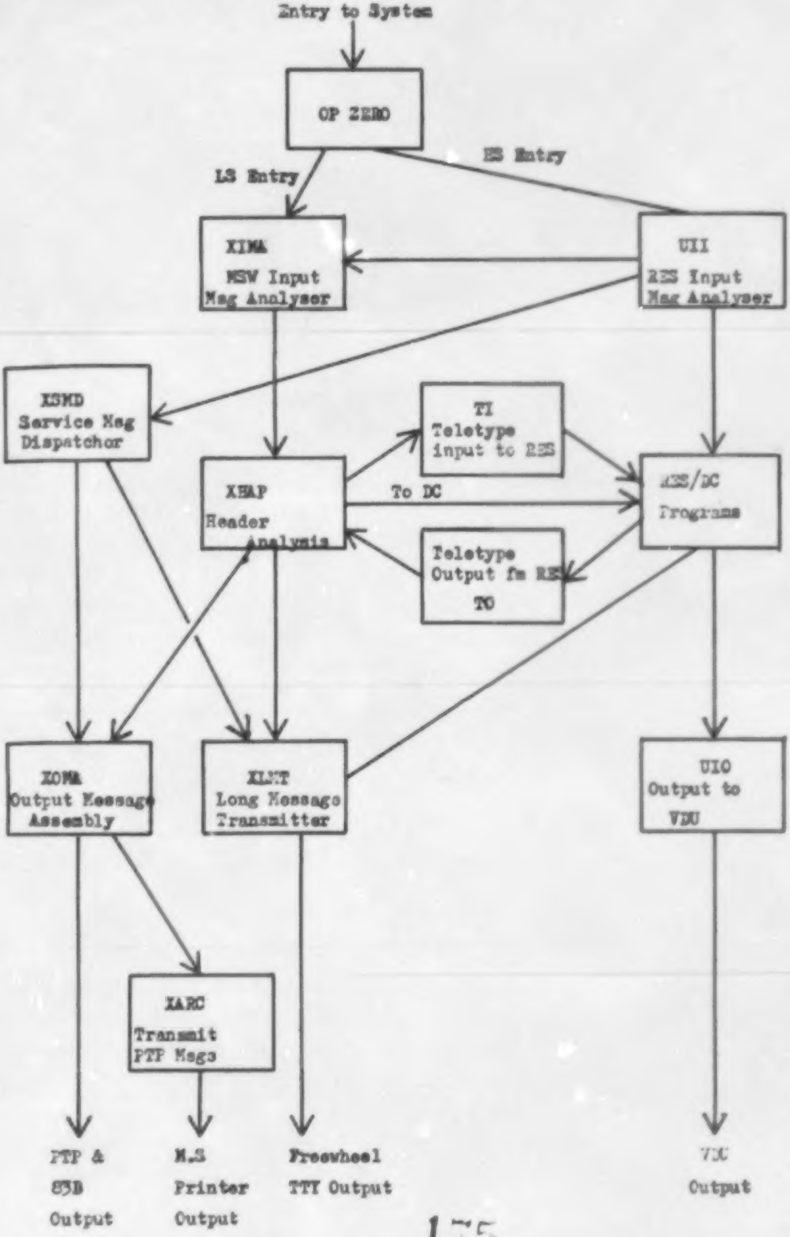
Queue Report by Priority contains the queue count in each priority level for the line or terminal indicated.

Airline Message Count contains the number of messages received from and sent to the various airlines.

From these reports are prepared the various costings for network sharing, and they create the historical information necessary to determine system growth and future circuit requirements.

5.4 LOGGING

All messages sent and received by the system are logged and held by the system to allow subsequent on-line retrievals. Periodically a purge is performed to clear the logs of references to the messages which have been transmitted or received up to a given day-time and the output messages are written to tape to enable them to be retrieved at a later date. This allows the file storage for these messages to be released for re-use.


Entry to System
OP ZERO
LS Entry
ES Entry
XIMA
MSW Input
Msg Analyser
UII
RES Input
Msg Analyser
XSMD
Service Msg
Dispatcher
TI
Teletype
input to RES
XHAP
Header
Analysis
To DC
RES/DC
Programs
Teletype
Output fm RES
TO
XOMA
Output Message
Assembly
XLMT
Long Message
Transmitter
UIO
Output to
VDU
XARC
Transmit
PTP Msgs
PTP &
83B
Output
M.S
Printer
Output
Freewheel
TTY Output
VDU
Output

PROGRAM PACKAGES FOR MESSAGE SWITCHING

5.5 CONTROL FUNCTIONS

Control is exercised over the message switching functions by use of various messages switching commands. These commands vary in length from three to six characters and all commence with the letter "Y". The use of these commands allows the following to be performed :

To hold or release the hold of traffic to a specified terminal.
To hold or release the hold of traffic to a specified line.
To monitor the traffic to/from a specified line.
To impose/release Alternate and Diversion routing.
To request the formation of system reports.
To change the tables which affect the running system. However, there are certain table changes which are not possible.

These fall into two classes : those requiring additions which would affect the rest of the table, and those requiring changes that would affect other tables. In such cases a new system generation would be required after a planned shut down.

To override the normal line control options for a line.
(eg. Address stripping, whereby addresses which are applicable to the circuit over which the message is being sent are stripped before transmission.)

Heading pattern generation and checking.

Ending pattern generation and checking.

5.6 CIRCUIT ASSURANCE

Circuit assurance is maintained by the use of sequence number checking and the transmission of half-hour check messages. The half-hour check messages are sent by the system to allow the terminal to maintain an audit trail of messages received. The system maintains its audit trail by comparing the sequence number received with that held from the previous message. Any discrepancy will raise an alarm. Additionally, the system will alarm if no messages have been received from a circuit for at least thirty minutes. In this way circuits without a protective line protocol can be monitored for possible outages.

In the event of terminal malfunctions or suspect circuit problems one of four test messages can be continuously transmitted on the circuit.

6.0 SYNCHRONOUS LINK

One of the most interesting parts of the IPARS system, from a communications point of view, is the trans-Tasman SLC link. It is interesting in the way a high-performance, dual priority, high integrity link is constructed using a basically simple voice-grade circuit by means of a good deal of complex programming.

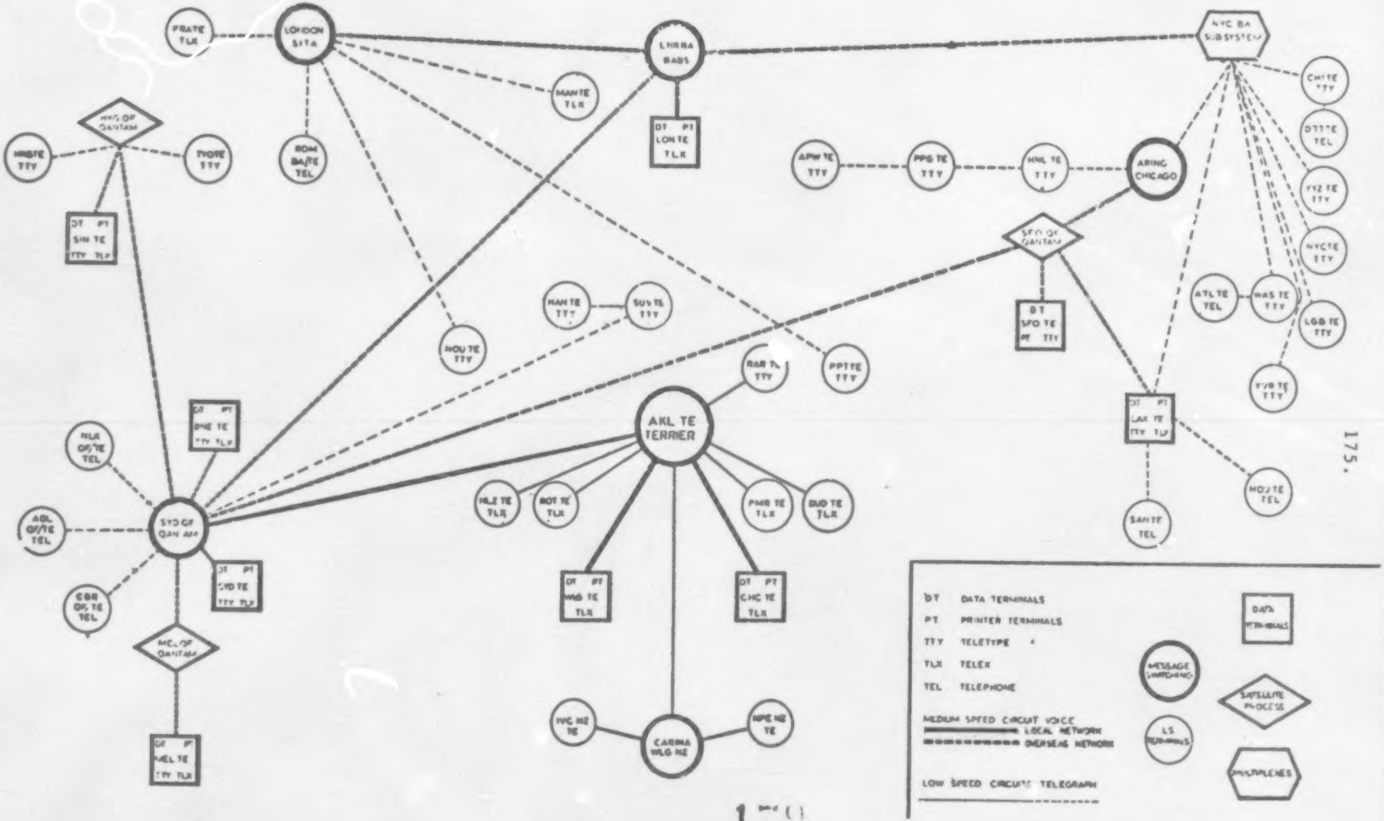
It is also interesting to view this link as if it were part of SNA (Systems Network Architecture) (Ref.6). Its Type A traffic consists of agent set messages which ultimately pass between a terminal and an applications program. Either the applications program is in New Zealand and the terminal overseas, or vice versa. At present, Reservations is the only applications program with overseas terminals, (Departure-Control being computerised only for Auckland Airport). CARINA-II reservations terminals are located in Australia, Singapore, HongKong, Honolulu, Los Angeles, San Francisco and London. All except the London terminals are connected to the QANTAM network. QANTAM communications programs drive all these terminals, but their messages are forwarded to CARINA-II for processing. The London terminals are driven by the British Airways Booking System (BABS) which forwards messages to New Zealand in two stages. They go first by SLC link to QANTAM, then on to CARINA-II. Conversely, there are Qantas and British Airways terminals in New Zealand which are polled by CARINA-II but whose messages are forwarded to QANTAM or BABS for processing.

It can be seen that a Type A message on the link is destined for one of three computers, and when it arrives it may be sent to either an applications program or a terminal. As these messages are, of course, all mixed in the data stream, each message has to carry with it its own routing identification. In fact, the envelope which surrounds each message, contains a HEX (High-level exit address) to indicate destination, a HEN (High-level entry address) to indicate source, and an indicator for Application vs Terminal. The tandem SLC links from London to Sydney to Auckland constitute the high-level portion of the OFTS network.

When an Air New Zealand terminal in London enters a message, the message traverses the local ALC line in London, then two SLC links to CARINA-II where it is processed. The return message follows exactly the reverse path, and ends up on the screen or printer of the agent in London. In spite of its long journey (through 4 SLC links and 2 ALC lines and the queues, if any, for use of these links/lines) the overall average response time, a few seconds, is small enough for satisfactory real-time operations by reservations clerks dealing with customers.

The architect of a communications link always has to strike a compromise between high performance and high integrity. In other words, he wants to send as much information as possible in a given time but must make sure none of it gets lost. If any is damaged or lost temporarily it must be retransmitted, and normal operation resumed as soon as possible. Control blocks acknowledging correctly received data blocks use up some of the available transmission capacity of the line, so the data capacity of the line is correspondingly reduced.

The SLC protocol deals with this conflict of objectives by placing a TSI(Transmission Sequence Identifier) in the envelope of each data block. A transmitting centre assigns TSI's serially, in a cycle of 31 possible values, regardless of message type. A receiving centre sends a positive acknowledgement control block


FRATE TLX
LONDON SITA
LHRBA BABS
NYC BA SUBSYSTEM
MANTE TLX
DT PT LONTE TLX
HKG OF QANTAM
HKGTE TTY
TYOTE TTY
ROM BA/TE TEL
DT PT SIN TE TTY TLX
APW TE TTY
PPG TE TTY
HNL TE TTY
ARING CHICAGO
CHITE TTY
DTTTE TEL
YYZ TE TTY
NYCTE TTY
SFO OF QANTAM
DT SFO TE PT TTY
ATL TE TEL
WAS TE TTY
LGB TE TTY
YVR TE TTY
MAN TE TTY
SUVTE TTY
NOU TE TTY
RAR TE TTY
PPTTE TTY
AKL TE TERRIER
DT PT LAX TE TTY TLX
DT PT BNE TE TTY TLX
NLK OF/TE TEL
ADL OF/TE TEL
SYD OF QANTAM
DT PT SYD TE TTY TLX
CBR OF/TE TEL
MEL OF QANTAM
DT PT MEL TE TTY TLX
HLZ TE TLX
ROT TE TLX
PMR TE TLX
DUD TE TLX
DT PT WLG TE TLX
DT PT CHC TE TLX
IVC NZ TE
CARINA WLG NZ
NPE NZ TE
HOU TE TEL
SANTE TEL
DT DATA TERMINALS
PT PRINTER TERMINALS
TTY TELETYPE
TLX TELEX
TEL TELEPHONE
MEDIUM SPEED CIRCUIT VOICE
LOCAL NETWORK
OVERSEAS NETWORK
LOW SPEED CIRCUITS TELEGRAPH
DATA TERMINALS
MESSAGE SWITCHING
SATELLITE PROCESS
LS TERMINALS
MULTIPLEXES

containing the TSI of the last correctly received data block. It does not need to acknowledge every block, but may allow up to 16 correctly received blocks before sending positive acknowledgement. Thus, provided positive acknowledgements arrive periodically, the transmitting centre may maintain an uninterrupted outward flow of data blocks. Receipt of a damaged block (parity error, format error, length error, etc) requires a negative acknowledgement. Receipt of a block with TSI not in sequence also requires a negative acknowledgement. Both cases are handled by immediately sending a control block containing the TSI of the last correctly received data block. The centre which originally transmitted the block must then repeat all blocks following the one so identified.

Messages greater than 250 characters in length are split into blocks of up to 250 characters. An MBI (Message Block Identifier) in the envelope of each block of a multi-block message indicates the sequential number (1 to 16) of the block within the message, and also applies a "label" to the block. Up to 7 unique labels are available, thus permitting up to 7 multi-block messages to be in transit in one direction at any time. With multi-block messages, not only must each block be correctly received, but the whole message must be properly reconstructed at the receiving end before a special control block (CML) is sent, authorising the transmitting centre to relinquish responsibility for the security of the message, and to release the message "label" for re-use.

As a footnote, it should be added that the most significant use of SLC protocol in the world is the high-level data network of SITA, based in Europe but will links to other parts of the world. The SITA network and OFTS network meet in London where they both have connection to the BABS system.

7.0 INSTALLATION AND TESTING

The decision to implement an automated reservations system in Air New Zealand was made in 1974, and from this decision began an investigation of those reservations systems available, to determine the most suitable. The selection of the British Airways Maxipars system provided a complete message switching package which meant minimal changes to conform with Air New Zealand requirements.

On receipt of the functional specifications a thorough check was made against facilities available and those required. Where discrepancies existed new functional descriptions were written for inclusion in the system. As the decision was also made to house the computer complex in a new building, the testing facilities were to be installed in what was the old computer centre in Air New Zealand House. This, because of space limitations, was restricted. For the Message Switching testing we installed a terminal interchange, eight VDU terminals and five medium speed printers. A test cell was also established containing one of the VDU terminals and two medium speed printers to act as the Network Control Centre, together with a selection of NZPO Olivetti low speed printers to cover the testing of half, full-duplex and simplex circuits. The NZPO also installed a patch panel to enable the configurations to be changed when necessary.

With this installation, testing commenced. Each aspect of the documentation was researched and a testing document was produced which detailed the various facilities available and the expected results under both normal and abnormal conditions.

Testing was then based on this document. Where the results did not comply with that documented, further research was done to determine :

a) Whether the documentation was correct

b) Whether further modifications should be done to ensure that the programs agreed with the test document.

This document was used for testing after each major modification.

Program testing was begun about September 1975, using CPSIM (Control Program Simulator). This test vehicle runs under OS/VS1 and simulates a number of the basic ACP functions, in particular the communications aspects. Messages are prepared as tape files of records in message format, and read sequentially to test single-thread processing. CPSIM controls and monitors the execution of the program under test, and produces diagnostic output on tape. A post-processor program formats and prints the diagnostics in a form readily usable by the programmer. Considerable usage was made of CPSIM, but mainly for testing Reservations and Departure Control applications.

Later, PTV (Program Test Vehicle) was used. It runs as a simple ACP system, simulating I/O only on certain nonimated lines. Multi-thread testing is possible, in fact, real I/O may be occurring concurrently on non-simulated lines. To simplify both CPSIM and PTV testing there is a utility program named STC (System Test Compiler) used to prepare files of test data.

However, with both Message Switching and the communications side of ACP, the major portion of testing was done on-line. For this purpose a small ACP system, known as the "Model Airline", was set up. It is still used now, even after cutover, for periodically scheduled test sessions. The model airline uses only 3 disk modules, and has only a few communications lines of each category. This was sufficient to drive the test cell equipment.

The usual pattern of testing was for a new program segment to be placed in the Real Time Object Library, and the previous version saved in a "fallback" area. An operation would then be attempted, resulting either in some form of output which could later be checked for correctness, or in one of a very large range of program errors which ACP is set up to detect, and which cause dumping of control tables, work areas, I/O areas, and currently used program segments. A considerable amount of testing involved the interface between the Reservations and Departure Control packages to ensure these packages produced in-format messages and were capable of receiving the same.

An infrequently-used but very powerful on-line testing tool is the Real Time Trace (RTT) which dumps tables and storage areas every time any program executes a co[illegible] program service call ("macro") of a particular class which is nominated beforehand by the tester. A more selective tool is Step-by-Step Trace (SST) which traces macros only in programs servicing entries from the terminal invoking the trace. Of all software testing aids, SST was found most useful by the communications and message-switching programmers.

A special software tool was used to some extent for testing the SLC link, namely Link Trace. After being started by an operator command, Link Trace writes a tape record for every SLC block sent or received, whether data or control block. The tape is later post-processed to print one line for each block, showing the time, the complete message envelope, the first five data characters (if any), and a selection of key control information. SLC software was actually in very good shape. Nevertheless, our test system was linked to an overseas computer whose link controller (the H632 front-end at Qantas) was using a different implementation of SLC protocol, and consequently Link Trace gave some valuable detailed knowledge of just what was happening on the link.

A second terminal interchange and terminals were installed to assist with the SLC link testing. The testing in this area was initially achieved by looping the link circuit back at the modem. This did nothing for message-switching as it made a mockery of the routing tables but was a great help in protocol testing. Later on, close liaison was maintained with both Qantas and British Airways to carry out two-way testing with Sydney which was then extended to three-way testing with British Airways in London.

At about this time we began using the Spectron Datascope. It was used on all types of line, but especially on the SLC link because there it was difficult to tie up particular messages to their related agent entries. This machine has enormous flexibility, and is the answer to a communications programmer's prayer (as well as having great use in the hardware trouble-shooting area). In brief, it taps into any line and displays all messages sent, received, or both, on a continuously scrolling screen. A knob is used to select display in hexadecimal, or whatever character set is known to be in use. The Datascope has a tape on which an entire test session can be recorded, then played back later, usually at slow speed, for careful analysis. There is also a data buffer (BDTS) in which a test message can be set up, either by keyboard entry (in hex) or by transfer from the tape. The data buffer is then discharged to the line, once or repetitively, in order to observe the response of the remote machine (be it a terminal or the ACP system).

Air New Zealand created their own test tool for one particular problem with some TTY equipment. This problem required the equipment to be driven by a much simpler program than ACP, so that the exact behaviour of the I/O interface could be observed. For this purpose the equipment was attached to a virtual CMS (Conversational Monitor System) machine, running under VM/370. By writing a rudimentary I/O handler and executing it under the I/O trace facility of VM/370, it was possible to pinpoint the equipment fault.

Over the testing period several thousand messages were prepared and entered either from the keyboard or on paper tape. A tape was prepared with a mixture of good messages and those with a broad spectrum of every error type that could be expected (and a few that couldn't). This tape was used at different stages of implementation as a base line test vehicle.

With the new computer centre at a stage where installation could commence, the equipment was transferred and testing continued. By this time, the cutover dates were rapidly approaching and the project moved into the final testing stages. These were the field trials, which entailed testing the system under actual operating conditions, using as much equipment as possible. As Air New Zealand was already using the network in a live situation it was impossible to extend the field trials to cover the total situation, so these trials were not as extensive as would normally be desirable. During this period the testing of all associated hardware being installed was carried out. This included the new circuitry, line and V24 switching equipment, the no-break and normal power distribution.

CUTOVER

The problem of cutting over from a real time system based in Australia to the new system in a new building with the added complication of a new International Terminal appeared to be immense when first considered. However, the planning for this event had commenced some eighteen months before. It was obvious that a "big bang" approach could not be taken. A series of cutovers extending over quite a long period with adequate settling down time between each cutover was the approach taken.

The first cutover was quite simple, being the shift of the main Auckland/Sydney and Auckland/Wellington links from Air New Zealand House to the Newton Data Centre. Apart from a few modem timing problems caused by the addition of additional modems into the circuit this was uneventful.

The next series of cutovers was to switch all terminals from being driven from the QANTAM Honeywell H316 concentrator to being driven through TERRIER. This involved changing the terminal interchange modem adaptors from 8-bit ASCII to 6-bit ALC and loading in the new program. This cutover was done a city at a time with a week between each. At the finish of this stage, TERRIER was acting as a concentrator, polling the terminals and transmitting the resulting data over the SLC link to Sydney and transmitting to the terminals the QANTAM responses. This gave an excellent indication of the TERRIER and the SLC link performances and with no problems experienced gave us more confidence for the cutovers ahead.

The next cutover was a major one, Message Switching, and took place on November 13th. It entailed the stopping of all low speed messages in and out of New Zealand, the change of routing tables in QANTAM, the severing of the double jumpering which the NZPO had done on all low speed circuits, and the bringing up of the new circuits required by the relocation of the Message Switching Centre to Newton.

The major communication aspects of cutover were now complete as the basic network was now established with all terminals connected to TERRIER.

The next major event was the transfer of the reservations records (PNR) from QANTAM to TERRIER. This transfer was to be done over the SLC link and consisted of approximately 25,000 PNRs. This imposed the heaviest loading that could be expected on the link, but again it was without incident.

The final event was the transfer of the terminal system from QANTAM to TERRIER and consisted of table changes within TERRIER and QANTAM to change the delivery points of the data. Again this was done by city with approximately a week's interval between each. The cutovers were finally completed on the 18th February, 1978. The Departure Control cutover, due to the inability to predict precisely when the International Terminal would be completed, was planned to slot in at any period of the cutover schedule after the message switching cutover. This event finally occurred on December 16th.

A summary of the cutover timetable is as follows :

20th August 1977	Relocating of the Auckland/Sydney circuit
23rd September	Cutover of Central Reservations Control
1st October	Cutover of Reservations Training Facilities
23rd October	First TI Cutover in Auckland
30th October	Wellington and Christchurch TI Cutovers
6th November	Remaining Auckland TIs Cutover
13th November	Message Switching Cutover
26th November	PNR Transfer
16th December	Departure Control
14th January 1978	Christchurch Sales offices from QANTAM to TERRIER
28th January	Wellington Sales offices from QANTAM to TERRIER
30th January	Hong Kong Sales offices from QANTAM to TERRIER
11th February	Sydney/Melbourne/Brisbane offices from QANTAM to TERRIER
18th February	Los Angeles/San Francisco/Singapore/London offices from QANTAM to TERRIER

8.0 FUTURE DEVELOPMENTS

As a service facility, the future development of the communications system is dictated by user requirements. However the increase of terminals in existing and new locations must be of prime consideration and inevitable as the company endeavours to remain competitive.

Other than such an expansion program there are various developments underway or being considered at the present time which include a communications aspect, such as :

Flight Planning : The introduction of computer flight planning into Air New Zealand has necessitated the development of a low speed circuit between CARINA-II ACP and the Commercial CICS system for the transfer of meterological data and completed flight plans to and from the message switching system.

Remote Polling : This is being considered for both low and medium speed circuits. Low speed to overcome problems associated with a multi-drop terminal system on an ARQ Radio circuit. Medium speed, firstly to enable direct control of our overseas terminal systems, and secondly to reduce the number of ports required at the 3705 for remotely clustered terminal sites.

It is obvious that what has been done for the airline passengers, in respect of immediate reservations must also be considered for the cargo consignees, and this has not been overlooked in development planning.

Also under active consideration are the changes to the networks which reflect back into the communications system. For example, the OPTS network is in the process of changing to the statistical multiplexor type of operation. To retain active participation in this network Air New Zealand must also be prepared to invest in this type of technology.

9.0 REFERENCES

1.	IBM Manual GH20-1473	ACP Concepts & Facilities
2.	IBM Manual GL24-3570	IBM 2946 Terminal
3.	Air N.Z. Manual IX-02-XFS	Message Switching Functional Specs.
4.	ATA/IATA document GEN/1840	Interline Communications Manual
5.	ATA/IATA documentAIRIMP	ATA/IATA Reservations Interline Message Procedures
6.	IBM Manual GA27-3102	System Network Architecture

A PROJECT IN COMMUNICATION: THE IMPLEMENTATION OF A NEW PROGRAMMING SYSTEM FOR NEW ZEALAND SECONDARY SCHOOLS

P.L. Collins and L.R. Hunt

The implementation in 1978, of a new programming system as a replacement for FORTRAN provides facilities for every secondary school in New Zealand to use a high level structured language on one of the ten Computing Centres operated by Databank Systems Ltd. The paper commences with a review of the use of the FORTRAN system since its introduction in 1971 and discusses the reasons for, and implementation of ALGOL-W as the first language on the new system. The additional features thus made available are fully described. Program development techniques are discussed and include design and implementation of an IBM/360 emulator as part of a "programmer's workbench"; selection of program structure and language for system components; tuning and performance; and the design of a suitable port-a-punch card. The paper concludes with a brief review of teacher training and programming assistance, other facilities available to secondary schools, possible extensions to the new system and suggested syllabus changes that could make the best use of these developments.

C.R. Categories: 1.51, 3.51, 4.12, 4.22, 4.42, 4.43
3.75, 4.6, 4.13, 4.31, 4.32, 6.21,
8.1

I BACKGROUND

PORTRAN [1], a dialect of FORTRAN developed at Otago University in 1969, has been increasingly used, since its introduction in 1971, to provide computing facilities for schools, on the Databank Systems Ltd. IBM computers at ten sites throughout New Zealand. Programs are hand punched into 'port-a-punch' cards (to avoid the formidable cost of a key-punch machine at every school) and taken to the trading bank nearest the school. The same afternoon, they are taken from the bank to the nearest Databank Computer Centre by the Databank courier. After overnight processing (with fixed job limits and fee) the cards and results are returned by courier, and are available for collection from the bank branch by mid-morning, regardless of the school location.

The Databank network, in conjunction with PORTRAN, has given New Zealand a unique national schools computing service, currently used by 256 secondary schools for classes of up to 30. However, PORTRAN provides only bare numerical computing facilities and in comparison with modern systems developments, appears progressively less attractive with the passage of time. PORTRAN usage rose from 6000 programs in 1971 to 39,000 in 1977. Associated with the increased use was increased expression of dissatisfaction, particularly as improved software become available elsewhere.

In 1975 a meeting was held to discuss computing in schools, and made suggestions for improving the system. A Department of Education Study Grant was made available to fund research into the problem, and Professor B.G. Cox of Otago University undertook to supervise the project. Databank Systems Ltd confirmed that the courier and processing arrangements could still be available for use with an enhanced system, and also offered development facilities and technical support.

In 1976 the Study Grant was awarded to one of the authors (Collins) and development of a new system was begun. At the same time, the other author (Hunt), on study leave from the University of Otago, was investigating computer teaching and language facilities overseas, while preparing the revised Computing Curriculum Bulletin [2].

II DESIGN CONSIDERATIONS

Gause [3], states that a fundamental principle of good design is that systems must be considerate of people. This principle was adhered to throughout the project. Hence, suggestions for systems features and options were sought from a wide range of people likely to use, operate or be affected by the proposed system, as well as those with experience in the design and use of similar systems. Schools using PORTRAN were circularised, a request for suggestions was placed in the N.Z.C.S. Bulletin, and enquiries were made among appropriate computing and teaching staff at various universities, schools and teachers colleges. The Department of Education, and the systems and operations sections of Databank Systems Ltd, were also approached for comment. Replies included:

Language Facilities:

extended character set, and string handling facilities;
recursive subroutine programming feature;
more data types and operators (e.g., bit, boolean, complex);
logical expressions and operators (e.g., <,>, NOT, OR);
dynamic array bounds, with more dimensions permitted;
mixed data structures and arrays of structures;
many standard functions, and provision for additions;
programmable exception control (e.g., endfile, overflow);
'block' structuring (e.g., as ALGOL, PL/I);
compile-time text replacement or 'define' option;
good diagnostic and program check-out features.

Batch Control:

global data files for class access;
flexible centralised charging based on facilities used;
file storage, recovery and editing commands;
user control of program limits (e.g., maxlines, maxseconds);
program cost analysis, by pupil and facility (e.g., lines, secs);
simple 'job control' and straightforward operating procedures.

Other:

a range of packages and languages to be available;
turnround to be fast, and charges modest;
standard 80-column cards to be accepted as well as port-a-punch;
good documentation and teaching aids to be readily available.

III THE SCHOOLS COMPUTING SYSTEM

The Choice of Approach

Several major decisions had to be taken before project development could begin. The main questions dealt with the logistics of program collection and delivery; data preparation method; type and location of computing machinery; choice of student languages; system structure; and development techniques, hardware and language.

The first questions settled were the location of machinery, and the logistics of program transport. It was found that the overnight Databank courier turnround could only be bettered by installing a computer or a remote terminal at every secondary school in the country, or by using the remote terminals being progressively installed in trading bank branches for communication with Databank Centre Computers. The cost of school-owned machines or networks was excessive and the trading bank network terminals were reported as lacking print facilities suitable for the style and volume of class program batch listing, as well as being scheduled for complete installation somewhat later than the project time-scale. The Databank offer of continued use of the courier network was therefore gratefully accepted.

The adoption of port-a-punch cards followed immediately as data preparation equipment for each school was far too costly, while centralised data preparation services would have critically increased both the operation costs and the turnaround delay. Card decks would be brought to the Databank Centres where processing would take place overnight. Their print set is only 48 characters, lacking most mathematical symbols, but this was accepted in the interests of speed, as transfer to 60 character sites would add another day. School work would thus be run on the IBM System/360 and System/370 machines under DOS and OS respectively, as low priority night-shift tasks.

A task handler was thus required that would run in the Databank environments, carry out the various batch control and other sundry requested options, and support a variety of programs and packages. A draft design for this was prepared and consideration given to confirming the choice of a first language.

The Choice of Language

Given the list of system features requested, there was some temptation to design a complete new language, especially adapted to the local needs and environment, but the project time-table made this impossible. The choice of a first language for implementation with the new system was therefore restricted to those with existing IBM 360 realisations that would be accessible to the project, and which fulfilled as many of the language feature criteria as possible.

Given the constraints, the suggested ALGOL-W was an obvious choice. Developed by Wirth and Hoare [4] to provide improved facilities in comparison with ALGOL-60, it incorporates most of the features requested for the system, including the extended character set. ALGOL-W is designed with the full 60 character set and it was necessary to truncate it to a 48 character set for this implementation, a restriction that would have occurred with any language running in the same environment. The initial development (at Stanford University) was for the IBM System/360 computers, and language extensions had subsequently been made, for the same hardware, at the University of Newcastle upon Tyne from which an offer of software and documentation was received. This offer was gratefully accepted, and permission sought of Stanford to modify and adapt their part of the software and documentation.

The Language Facilities of ALGOL-W [5]

The language is 'block structured'. Blocks can stand in place of any 'simple statement' and are delimited by 'BEGIN' and 'END' All variables must be declared at the start of the related block, and are unknown outside it. This 'constraint' is actually of particular advantage to naive users. It helps insulate them from the effects on their programs of otherwise undetected spelling or punching errors. Memory allocation is dynamic and storage is assigned to variables declared within a block when the block is entered and released when the block is left. This mechanism applies to blocks both embedded within control structures (e.g. IF...THEN <block> ELSE <block>) and forming the bodies of procedures (e.g. PROCEDURE <name >(<parameters>); <block>).

Procedure and function declarations and statements may be fully parameterised. Properly used this can completely remove unpredictable side effects from large sections of a program's code. Programming of procedures is streamlined by embedding in the parameter list. E.g., "PROCEDURE <name> (REAL VALUE RESULT A,B,C)". "VALUE" and "RESULT" options may be used separately or together to improve program run-time efficiency, to document the programmer's intentions on 'data flow' direction, and to enforce those intentions with built-in checking mechanisms.

All declarations may use variable names up to 255 characters long. This is quite enough for very descriptive identifiers. Variables may be of many types, either simple or grouped. Simple types are INTEGER, REAL, COMPLEX, LOGICAL, BITS and STRING. LONG is available, but only for REAL and COMPLEX types. Collections of data under group identifiers may be ARRAYs of the above types, or more comprehensive groupings termed RECORDs. These are collections of heterogeneous data types similar to the mixture that may be placed on a standard data file. However, in ALGOL-W the term RECORD does not apply to a file element but to a dynamic internal storage unit which is created, with an attached pointer (termed a REFERENCE) whenever an assignment is made to any part of an unallocated RECORD.

The form of RECORD/REFERENCE correspondence allows programming with data structures, lists (rather than arrays) of structures, arrays of pointers to structures, and linked lists and trees.

ALGOL-W limits program errors resulting from "default" data type conversion rules which apply in some languages. For example the assignment of a real variable to an integer variable is not possible without the use of a built-in function like ROUND or TRUNCATE. As well as the usual arithmetic operators, there are operators provided for handling LOGICAL, STRING and RECORD variables. These include comparison e.g. < (less than) and the boolean operators NOT, AND and OR. For program control structures ALGOL-W provides the conditional statements IF and CASE, and the repetitive statements FOR and WHILE.

The compiler has a trapping mechanism for exceptional conditions such as end-of-file, divide-by-zero, overflow, underflow, square root error, etc. The programmer is provided with a number of options which enable such exceptional conditions to be replaced by appropriate limiting values (e.g. ± MAXREAL, 0, SQRT(ABS X), etc.) for a specified number of times, and for an (optional) error message to be printed each time. The compiler also provides tracing and debugging options for run time diagnostics. These include a post-mortem dump of variables, flow summary and store tracing. These options are all helpfully named and formatted.

A Cautionary Tale

The first step in preparing ALGOL-W for inclusion in the system was the preparation of a DOS interface for this compiler, originally written for an OS environment. In concept this was not too difficult, and the required programming was duly completed, assembled, and placed on the system library.

Next, the addition of 48-character set equivalents to the ALGOL-W 60-character text recogniser was planned, to conform to the Databank 48-character print set.

ALGOL-W is written in PL360, which was available to us in the package provided from Newcastle, and which would run well on the Dunedin Databank Centre IBM 360/40. Unfortunately, however, PL360 is also a 60-character language, and its output was not intelligable using the commercial print set. PL360 could not have been converted to 48-character set by amending and recompiling it at Databank, because it is itself written in 60-character PL360.

Several months were spent carrying magnetic tapes of print files from Databank to the University for printing on the Burroughs B6700 60-character printer, carrying update tapes and cards in the reverse direction, and waiting for the very patient Databank operations staff to 'create' additional development time. The extra travelling, programming complications and scheduling difficulties rapidly multiplied into an administrative nightmare.

When it became clear that the situation was worsening with the passage of time we sought, and failed to find, alternative development accommodation with the necessary facilities within a reasonable distance of the project base at Otago University. Finally, rather than abandon the project entirely, it was decided to write a complete IBM 360 hardware emulator to run on the university Burroughs B6700, and to use that for system development and initial testing, with the Databank facilities to be used only for advanced testing and implementation.

Six months later the "programmer's workbench" [6] was complete (described in a later section) and had passed its testing with a 'sysgen' version of DOS 26 generously made available by IBM New Zealand Ltd. The original project was then able to continue, back under control, with programming, processing and printing all on one site.

The Task Handler

This is a group of routines, written in IBM 360 Assembler and PL360, to read mixed port-a-punch and EBCDIC card decks, convert the whole input stream to EBCDIC card images, provide text define and expansion facilities, create and merge global file copies into the data stream, initiate the user's choice of compiler or package, pass the generated EBCDIC card images to it for processing, and maintain the task costing and summary tables.

Most of the system enhancements are based on the card design described in the next section, and are processed by a PL360 procedure written as a 'finite-state automaton' [7]. This is one of a class of table-look-up text recognisers, characterised by rapid operation, simplified memory management requirements, and ease of design (by algorithmic process) for various purposes generally useful for syntax analysis or text editing. Because the automaton is controlled by a pair of simple numeric tables (the 'action' matrix which tells the system what to do next - and the 'state' matrix which notes what the next input item must be in order that it be accepted as valid) the functioning of the handler can be significantly altered or extended by doing little more than changing or adding to the arrays of table values.

The automaton, and any user-selected program that follows it, are run as separate task overlays under control of a 'root phase monitor' that is nominally core resident throughout the batch. Acknowledgements are due to Stanford University for some of the control techniques, adapted from the ALGOL-W procedures for this purpose.

Since the monitor was first implemented with the preprocessing automaton it has been further extended to give batch costing summaries, and it is shortly to be modified to provide a central accounting service, with the debits based on the cost of machine facilities actually used, rather than on a fixed fee per program.

196

The Card Design

The card, illustrated below, is partially pre-perforated by die-cutting during manufacturer, to allow 'chads' to be manually punched out with a paper-clip. To avoid inconveniencing computer operators with card reader data-checks the design requires punching only one chad per column, and this is stressed in the teaching documentation now being prepared.

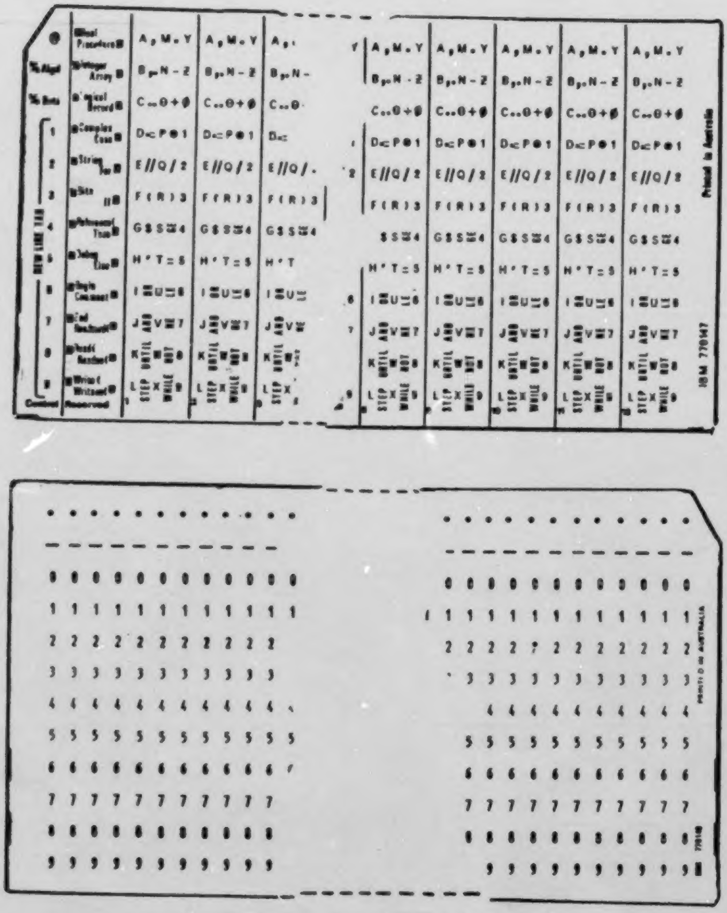

Procedure
Integer Array
Record
Complex
String
Extended Type
Line
Comment
Control
Reserved
NEW LINE TAB
A , M . Y
B , N - Z
C .. O + ∅
D ≤ P ● 1
E // Q / 2
F (R) 3
G $ S ≡ 4
H ' T = 5
I ≡ U ≡ 6
J AND V ≡ 7
K UNTIL NOT ≡ 8
L STEP X WHILE 9
Printed in Australia
IBM 770147
0 1 2 3 4 5 6 7 8 9

Figure 1. The Card Design

To maintain the mechanical strength of the card, only the alternate columns of the card are die-cut. The other columns cannot be used. Depending on which way the card is fed, either the 'even' or the 'odd' card columns may be punched. This fact is used to allow a 'double-sided' card with two different interpretations. One side is restricted to numbers - optionally signed integers or real values. The other side has alphanumeric and special symbols.

The alphanumeric side of the card is in three parts. The first provides several 'job control' features, or may be used to force a new line with an option of fixed tab indentation. If no tab or other control column feature is used, translated card images are concatenated and overflow to column 1.

The second part of the card generates reserved words from either 'side' of the field, or both. The words chosen for inclusion here are those thought most likely to be used at the left of a line of text.

The third part contains twelve identical fields. Each has three columns of chads, with the alphanumeric character set printed over the chads. Between these are printed two columns of special character combinations and reserved words which are selected by punching the bracketing chads. For example, 'BN' results in ',.'. A further feature is that punching one chad from each of the 'M' and 'Y' columns (but not necessarily on the same line) results in one of the 144 reserved words or character combinations provided. For instance, 'QB' results in 'LONGARCTAN('. The list of reserved words and associated chad pairs is included in all user guides for the system.

An equivalent feature is provided for the 'A' and 'M' columns, with the user able to punch his own text for inclusion whenever that chad combination subsequently appears. Such 'user defines' may be 'global' and apply to all programs in the batch, or they may be 'local' and apply only to the single program in which they are set up. This allows function inclusion and descriptive variable names of any length to be generated by punching only two chads.

IV THE PROGRAMMER'S WORKBENCH

Background

The educational system was originally to have been developed on Databank Dunedin Centre IBM 360/40, but this proved impracticable, for the reasons given earlier, and an emulator, allowing IBM 360 object code to be run on Burroughs B6700, was eventually implemented, and used for all subsequent systems development.

Although the completed emulator actually ran at much less than half the speed of the actual hardware it models, this shortcoming was more than compensated for by the convenience of having available the appropriate character set, and working with a familiar keyboard and text editor. Ivie [6] suggests that our experience (working better for a new machine from within a familiar environment) would probably be a general one, and has proposed a "Machine for Software Development". We certainly endorse the suggestions in that paper.

As well as providing a familiar work environment, emulating brought other benefits. Among the most significant of these were the problem analysis and tracing routines that were written into the emulator. These tools helped considerably in the finding and solving of quite complex bugs. An example: during production of the DOS interface a programming error was introduced into the DOS ALGOL-W trace code. Tracing was by far the best technique to solve the problem, but the provided trace could not be used for this, as it was itself faulty. In any case even a perfect tracing routine becomes infinitely recursive if monitoring itself. Hence, only with the separate trace in the workbench emulator could the fault be quickly corrected.

The Emulator

This program, written in Burroughs B6700 Data Communication ALGOL [8], mimics the hardware characteristics of the IBM System/360 CPU and peripheral units [9].

The basic functioning of the program is actually quite simple. It 'performs' an Initial Program Load by reading from a nominated file (for an emulated 'device') into a program array 'CORE' and then interpretes the contents of that array as IBM 360 operation codes and operands. Operations are 'fetched' one at a time, identified and executed as specified in [9].

Emulation of system interrupts is done by 'causing' B6700 events to which software interrupts have been attached. I/O functions are for the most part carried out asynchronously, using the B6700 software interrupts to set the semaphores that will pass the status of the emulated device to the emulated CPU when the emulated interrupt becomes unmasked and is taken by the actual operating system running in the emulated machine. This may seem confusing, but once the emulator was implemented it became unnecessary to consider such distinctions as it could then be viewed, programmed and operated just as if it were a hardware machine of the model being emulated.

Emulated 'core size' is limited in the present version to 384K bytes by a decision to forgo generality in favour of speed and simplicity - using only a single level of subscripting to address the array 'CORE' considerably improved performance while still giving a machine limit quite large enough for the required testing environment. Fortunately, the overlaying philosophy of the B6700 hardware and software relieves the programmer of any need to consider the actual size of the host machine, which is able to 'contain' the emulation of hardware many times larger than itself simply at the price of some overlaying overhead.

Disk Pack Emulation

The emulated 'disk subsystem' unlike the emulated core, could have been severely constrained by the host (B6700) configuration. Only one disk-pack drive was available for emulation runs, yet at least three emulated drives were needed if the jobs tested on the emulator were to be allowed to run with completely unmodified input streams. This was required to allow testing job control decks as well as programs - besides making test procedures and administration very much simpler. To accommodate such requirements we implemented "virtual diskpack" with many emulated full-size disk packs resident in one actual diskpack.

The "virtual diskpack" system exploits the IBM 2311/2314 data organisation philosophy which does not divide disk tracks into blocks, buckets or segments, but treats each track as a continuous area into which variable length records may be effectively concatenated until the track capacity is exhausted.

Each of the records is prefaced with control and length information. (Fixed length records are merely a special case of the more general facility.) Because of the above data organisation, each work pack, prior to use, may be completely described by the set of short track header records ('RO') for the empty tracks and requires much less than 1% of the actual pack volume for its description. By organising the emulated tracks in inverted list files of actual variable length track records with a concatenating garbage collection system it has proved possible to recover actual disk space relinquished by each emulated disk when it writes 'short tracks' so that the freed area becomes available to the other emulated packs sharing the same actual spindle.

Inter-task Communication

An interesting aspect of the emulator program is that it needs another program to control it, because otherwise its 'operator' has no way to 'press' the emulated buttons, which are, after all, only values held by variables within the emulator program.

The emulator has a control monitor program which co-exists with it, in the B6700. The monitor is run at a higher priority than the emulator to ensure that it will be able to gain control when this is required. It provides the operator interface with the emulated machine, and has its own language in which the user specifies what hardware actions or options he wants. Examples of the terms used are: DEVICE X'291'='2314'; READY X'180'; RESET UNITS; CONFIGURE; TRACE DEVICE X'00C'; TRACE INSTRUCTIONS ALL; RESET TRACE; SAR CORE X'50DC'; SAR INSTRUCTION X'56'; RESET SAR; STEP; STEP 25; IPL X'291'; IPL; REQ; EOB; POWER; OFF; DIE.

The monitor intercepts emulated console messages in both directions and matches the characteristics of the IBM 1052 console with those of the Burroughs remote TD800 display unit which provides the emulator user with his interface to the system.

A 'real-time' clock is emulated. It keeps accurate real time, not emulated time, and is updated approximately once per second by a programmed B6700 timer interrupt which also gives the monitor control, briefly, to check whether an emulator timing interval has expired, in which case the appropriate emulated external interrupt is set.

V IMPLEMENTATION

Pilot Implementation

A test version of the system, supporting only ALGOL-W, was first implemented under DOS at Databank Dunedin Centre in November 1977 in conjunction with a pilot course in ALGOL-W for school teachers. During the first term of 1978 the pilot system was extensively used by six schools in Otago.

In May 1978 an upgrade was released. It added support for global files, defines and batch costing. The release was initially at Ellerslie and Hamilton Databank Centres to coincide with ALGOL-W courses presented in those areas by Otago University on behalf of the Department of Education.

The system is now in routine use at all Databank DOS installations and over 1000 ALGOL-W programs have been run under its control, by the selected pilot schools. The authors have prepared teaching material in draft form and this is being progressively refined in the light of experience being gained with it at ALGOL-W courses.

Projected

Further courses for teachers are planned for August 1978 in Wellington and Auckland and a final version of teaching documentation is expected to be available for national release shortly thereafter. It is also hoped that the Department of Education will subsequently be able to provide its own regional and local courses in ALGOL-W.

An OS/DOS interface version of the system is under test at Databank Wellington Centre, and its release at the OS centres will complete the national implementation of the system.

Scheduled development includes centralised national charging and further improvement to the 'readability' of ALGOL-W error messages and diagnostics. Routine system support is being continued throughout the introductory period. At present (May 1978) there are only six unsolved items on the Trouble Report List - none of them serious.

By the end of 1978 implementation of the system with ALGOL-W as a first language will be complete. There will be a number of teachers at each of the main centres who have a reasonable degree of expertise with the system and it will be available nationally as a standard teaching tool from the beginning of 1979.

Testing and Implementation Procedures

Test schedules are based on a three step sequence. Firstly the effective working of any new features is established with specific test data, on the workbench machine. Tests include a reasonable selection of predictable correct and incorrect input for each tested feature. Correct output is of course required from correct input, and non-fatal response is expected from incorrect input. The more likely an input error is, the more helpful the response is required to be. Errors in one program within a batch must not affect other programs in that batch.

The second test step is the preparation of a system release tape, which is then tested on the workbench against the standard benchmark. This consists of a wide range of user programs giving adequate cover of what is hoped to be the full range of system features.

The third step consists of a repetition of the benchmark test, carried out at the Dunedin Databank Centre, and followed by pilot implementation while schools in the Otago area co-operate by trying to 'break' the new features.

Finally, after several weeks of user exercise, a distribution tape is forwarded to Databank Head Office for national release.

It is of course impossible to test exhaustively. Test oversights may be random or due to bias. One particular failure was quite revealing. After conversion of ALGOL-W to 48-character set the system was left with a bug which would cause all "GO TO" statements to fail. Readers may draw their own conclusion about our personal programming technique, and the style that we teach, from the observation that not only was the error detected by neither of us during testing, but also that hundreds of programs were run during and after the first training course before we received a Trouble Report on the fault. We acknowledged the fault but have not yet fixed it. The present implementation is therefore of a language without a GO TO. Its rapidly increasing acceptance in this form probably points up some sort of moral [10].

Performance and Tuning

The pilot version of the preprocessor was written in a subset PL/I using recursive descent, a 'top-down' type of programming technique which can produce slow programs if 'CALL' carries substantial overheads. This is unfortunately the case with some PL/I implementations, so it was disappointing, but scarcely surprising, to find that the port-a-punch cards were being processed, albeit correctly, at a rate of about one per second.

A five-fold speed improvement resulted immediately when the preprocessor was programmed in PL360 as the finite-state automaton described earlier. The new version, when it had passed the essential acceptance tests, was exercised with input streams of 'typical' batch ALGOL-W programs, and detailed analysis of its control path undertaken with the aid of the workbench programmer aids. The more frequently threaded code segments were quickly identified, the programming techniques used on those paths were reviewed, and alternatives were considered. Finally a five line patch was designed and implemented, leading to the speed improving by a further factor of four. The preprocessor then became I/O bound to the card reader and further tuning was held over until preparation of the OS version was complete and ready for final testing.

The actual effect of preprocessing on running costs is not yet known. Preliminary estimates are that a typical ALGOL-W program might cost between 10 and 40 cents to compile and execute, depending on the facilities used, and that preprocessing would account for between 10% and 35% of that. At current charges and volumes the cost of a second per program is about $500 per year. It is therefore worthwhile to consider action to save several seconds per program. However, analysis to determine whether such improvement is feasible will be held over until the charging algorithm has been fully tested and implemented. The interim costing procedures are known to make inappropriate allowance for non-chargeable overheads and are therefore unsuitable for detailed performance measurement. No charge will be made for schoos use until the charging algorithm is functioning correctly.

VI DISCUSSION

A steady increase has been continuing in the use of PORTRAN since it was introduced eight years ago. However, perhaps because of the limitations of PORTRAN, not all schools use this facility. Some schools send FORTRAN and BASIC programs, by courier or mail, for processing at a university computer centre. Others have private arrangements with various companies, and several schools have their own minicomputers.

The new system is not a programming language, but an interface between port-a-punch cards and other software. The continued use of port-a-punch cards is perhaps unfortunate, but it remains one of the best options. We are convinced that with the addition of ALGOL-W and the global file and text expansion facilities it has become vastly superior to PORTRAN, FORTRAN and BASIC both for teaching and for general programming.

The current release of the system does not support source text libraries or editing facilities, object libraries, or disk and tape file handling. All these options can be readily included, but perhaps there are policy questions, concerning central storage of school files, that should be answered first.

Other compilers can readily be added to the system. The card design has a further 1820 vacant define combinations available, and the system design philosophy has aimed at the necessary modularity to allow ready extension. Packages and utilities can also be linked for schools use in a similar manner, and could readily embrace not only academic functions, but also administrative, such as mark scaling and timetable planning.

The needs of subjects other than numerical analysis can also be catered for by this system. Suitable options can be programmed in ALGOL-W or added to the system as packages. These could include simulation of world or local population dynamics and food distribution, business and profit modelling, social studies research analysis, programming exercises in accounting, language text analysis, and the usual applications to the physical sciences.

We would hope to see the current advances in computing facilities lead to curriculum changes that will encourage more representative teaching, and a more balanced understanding of computers and computing in New Zealand schools.

VII ACKNOWLEDGEMENTS

This project has been funded by the Department of Education. Databank Systems Ltd have donated substantial amounts of machine time and are assisting with the DOS to OS conversion. Professor B.G. Cox has supervised the project and Mr D.R. Forster provided much help, encouragement, and technical advice. We would also like to acknowledge Stanford and Newcastle-Upon-Tyne Universities for the ALGOL-W software and documentation, and IBM (NZ) Ltd for the DOS/OS system software and documentation. Our thanks are also due to colleagues, teachers and pupils who have contributed in various ways to the project.

VIII REFERENCES

1. James, N.S. FORTRAN, Fourth Edition, 1973, John McIndoe Ltd.
2. Computing - A Revision of Curriculum Bulletin No. 72, Computing Section 1978, pp 1-96.
3. Gause, D., Soft Designs for Hard Systems, Seminar, Dunedin, 1976.
4. Wirth, N. and Hoare, C.A.R., A Contribution to the Development of ALGOL, CACM, Vol. 9 No. 6, June 1966, pp 413-432.
5. ALGOL-W Programming Manual, 1974, University of Newcastle-Upon-Tyne.
6. Ivie, E.L., The Programmer's Workbench - A Machine for Software Development, CACM Vol. 20 No. 10, October 1977, pp 746-753.
7. Lewis, P.M., Rosnkrantz, D.J., Stearns, R.E., Compiler Design Theory, Addison-Wesley, 1976
8. Burroughs B6700/B7700 DCALGOL Reference Manual No. 50000052, June 1973.
9. IBM System/360 Principles of Operation, 9th edition, 1970, GA22-6821-8.
10. Dijkstra, E.W., GO TO Considered Harmful, CACM Vol. 11 No. 3, March 1968, pp 147-8.

A FRIENDLY TERMINAL - WHY NOT?

AUTHOR

PETER COOKE
Senior Systems Engineer
IBM New Zealand Ltd.

ABSTRACT

More and more people whose prime concern is some specific business activity are and will be directly and casually using terminals.

No matter how clever the programming, it will be factors such as ease of use at the terminal, difficulty of training and proneness to errors that will determine the user attitudes that finally make or break the system.

A well designed system with a suitable interfacing dialogue at the terminal, is therefore a key requirement for any successful on-line application.

This paper explores such design considerations, with emphasis on commercial systems.

KEY WORDS

terminals, system design, system's menu and dialogue, data entry, operator response, display format, on line editing.

C.R. CATEGORIES

2.1, 2.12, 3.50, 3.79, 3.80, 4.49

A FRIENDLY TERMINAL - WHY NOT?

PETER COOKE
Senior Systems Engineer
IBM New Zealand Ltd

ABSTRACT

More and more people whose prime concern is some specific business activity are and will be directly and casually using terminals.

No matter how clever the programming, it will be factors such as ease of use at the terminal, difficulty of training and proneness to errors that will determine the user attitudes that finally make or break the system.

A well designed system with a suitable interfacing dialogue at the terminal, is therefore a key requirement for any successful on-line application.

This paper explores such design considerations, with emphasis on commercial systems.

1.0 INTRODUCTION

1.1 THE WORLD AROUND US

We are living in an age of rapid change. The consumer market is being saturated by complex products, such as pocket calculators and digital watches at costs that make replacement cheaper than repair. Global colour television transmission of all important international events is taken for granted. Children are growing up accepting that man has always been in space and able to visit the moon.

The results of technology (for better or worse) are all around us.

One of these results is the computer ...

Each day sees more and more people being affected in their daily lives by computer systems. Computer systems which have an ever-increasing number and variety of transactions to process concerning us and our business affairs.

What is happening is that applications are changing to more directly encompass the commencement of a transaction. By placing a suitable (often specialised) terminal at the worksite itself, the terminal becomes an accepted tool to be used as and when needed on a demand basis. Like many other facilities, such as the telephone, its convenience and low cost will be the prime considerations and not the number of keystrokes or the device utilisation.

The user will have his or her responsibilities primarily aligned to the activities taking place at the worksite and will have little or no allegiance to or interest in Data Processing as such. Most terminal usage will be on a somewhat casual basis and the user will not desire, nor should in many cases require extensive training on how to use the terminal system.

As terminals proliferate at department stores, at banks and other places, it is inevitable that many employers and a significant proportion of the general public will become aware of the presence, not to say intrusion, of the computer system (as represented by the terminal) into their business or leisure activities.

When speaking a great deal of information is conveyed since 'voice' is a communication-rich medium of communication. Any approach based upon typing takes longer and is significantly less effective. (1) It is this recognised less effective mode that is the basis of the man-to-system interface for most commercial systems.

Even if no other reasons existed, this is surely reason enough to want (need!) to study those factors that make a computer system easier to use.

2.0 MAN-TO-SYSTEM INTERFACE

2.1 THE IMPORTANCE OF THE MAN-TO-SYSTEM INTERFACE

While most aspects in the design of an on-line computer system are important, the interface presented to the user is one of the most critical. It is here that the usability and usefulness of the system is judged and perhaps found wanting.

To appreciate this, it is necessary to understand the interactions and feelings that a user develops as a terminal system is used. The user's intention is to carry out some particular task using the terminal. Typically, this will involve a number of interactions with the system as the desired request is specified and all required data identified or entered. A dialogue can thus be said to take place between the user and the system - a dialogue which requires at certain stages very intense concentration by the user.

For the user, a sudden variation or disruption from normal during the period of concentration is traumatic. It must be realised that a very strong emotional relationship builds up and this must be taken into account when planning the system.

---

(1) Reference 4

A user can become very frustrated and dissatisfied if required to cope with a clumbsy, complicated and seemingly illogical dialogue. For this reason the dialogue should address the most obvious cases in a natural way leaving any complexity to become apparent only for the less frequent cases.

The best time to get the dialogue right is at the beginning, since a poor start with an on-line system is very difficult to overcome once the users have lost confidence, lost interest and withdrawn their support.

2.2 TERMINALS

Terminals are becoming more specialised as the applications extend outwards to where the transactions originate. Some of these terminals have been uniquely developed with a variety of reading wands and scanners, while others were the evolution of existing devices such as cash registers. As a consequence, a wide variety of media such as optical characters, bar codes, magnetic stripes etc can be created and accepted.

Today and for some time in the future, the main work horse is and will be based upon the keyboard-display unit. Currently implemented with Cathode Ray Tubes as the display device they are known as CRT's, Visual Display Units or VDU's. Gas Plasma Displays are, however, emerging as an alternative display device for some applications.

The emphasis within this paper is on interactive commercial business systems and the use of a VDU-type device with an alphameric keyboard and display is assumed.

2.3 SOME REQUIREMENTS

The earliest on-line systems involved well defined transactions and specially designed terminals which were operated by people trained in their use. Using the terminals was, in fact, an important part of their job.

Today, computer-based applications must interface with operating people who are busy with their regular duties which are in the main stream of company day-by-day activities. For these people to use the system successfully, the terminal and its interface must be :

- Easy to Use
- Not distracting from the primary job
- Reliable in the job environment

This leads to the following requirements :

1. The actual terminal must be safe, easy to use and not cause operator fatigue.

2. The response time must be suitable for the particular activity.

3. The operator needs to be kept informed and aware of current status and valid options.

4. Guidance information needs to be readily available at all times.

5. The operator must not be overwhelmed or made to feel inadequate or dominated by the system.

6. The dialogue must appear natural and straight forward for the great majority of situations. Its design should not be compromised in order to cater for infrequent odd-ball circumstances.

3.0 SYSTEM & DIALOGUE DESIGN CONSIDERATIONS

3.1 BASIC ACTIONS FOR ENTERING DATA

3.1.1 Fundamentals (or stating the what-should-be-obvious)

Before commencing the design of any dialogue it is necessary

BASIC ACTIONS

DATA IDENTIFICATION

FIGURE 1A

INTENT	TECHNIQUE		EXAMPLE
Indicating the identification of the data items being entered			
	Positional		
	- Free Form with Delimiters		Fig 2A
	- Fixed starting position		Fig 2B
	- Fixed Form		Fig 2C
	- Question & Answer		Fig 6
	Self Identifying		
	- Keywords or other specific field-identification directly associated with data		Fig 2D Fig 2E Fig 2F Fig 2G
	- Selection (eg. lightpen) followed by data entry		Fig 2E
	- Use of special keys which may also serve as delimiters		Fig 2F
Delimiting the data items			
	- Fixed Length Pre-Allocated		Fig 2C
	- Delimiter(s) used (Tab Comma & blank are frequently used)	These could be special keys or symbols that also cause field identification to be assigned.	Fig 2A Fig 2F Fig 2G
Signalling the end of the entry			
	- Enter Key or Motor Bar		
	- Other Special Key(s)	These could be the special keys that also cause field identification to be assigned.	
	- Character Count		

BASIC ACTIONS

DATA ENTRY

FIGURE 1B

INTENT	TECHNIQUE		EXAMPLE
Actual entry			
	- The actual data is keyed	It can be new data, or	Fig 2A
		an amendment or replacement of existing data which is being displayed.	Fig 2I
	- Acceptance of a default		
	- By meanings assigned to special keys (may also signal end of entry)		

to be very clear about what has to be communicated explicitly or by inference as items of data are entered. By one means or another the editing logic must know or deduce:

1. The idenfification of each item of data being entered.
2. The starting and ending positions of each item of data.
3. That the actual entry of data has reached such a stage that processing (editing etc) can commence.

A number of approaches have evolved and these are summarised in Figure 1A. Figures 2A to 2I show typical screen layouts and these should be studied carefully.

3.1.2 Actual Data Entry

Data is usually entered from a suitable keyboard. However, in many instances, it is possible to predict the most likely values to be entered, in which case, the keying effort can be significantly reduced by suggesting these as a default entry. Figure 1B summarises data entry techniques and figures 2H and 2I provide examples.

3.1.3 Positional Input

With positional input each data item is interpreted according to its location relative to defined points (such as start of input) or to other data items. Unless the data items are all of a fixed size in their normally expressed form the variable length form is far more natural. People think of numbers such as twelve as '12' and not as '00012'. We should therefore set out to always use the most natural and convenient way of expressing the input. This may involve a little more programming effort but leads to a more acceptable, more tolerable interface.

In written expression a blank separates words but multiple occurrences of a blank have the same significance as one blank. Provided a blank was not permitted within the data,

INDICATING THE FIELD IDENTIFICATION

A : BY THE POSITION OF THE ITEM OF DATA

```
ENTER CUSTOMER #, ORDER DATE, VALUE

    123456, 11/11/78, 168.40
```

Fig 2A

Positional - (Free Form)

Prompt given.

Free form input with , as delimiter.

Suitable for variable length input.

```
CUSTOMER # ........................

ORDER DATE ........................

VALUE      ........................
```

Fig 2B

Positional - (Fixed starting position)

Each field starts at a fixed position.

Variable length fields.

TAB key used as delimiter and to advance to start of next field.

```
ENTER REFERENCE NUMBER R

     ORDER DATE DD/MM/YY

     VOUCHER NUMBER V

RRRRRR    DD/MM/YY    VVV

123456    11/11/78    789
```

Fig 2C

Positional - (Fixed Form)

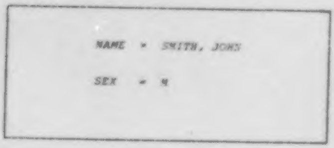
Prompt given.

Fixed form.

Data replaces field indicators or is keyed below.

Suitable when input has natural fixed length.

B : BY FIELD IDENTIFICATION ASSOCIATED WITH THE DATA



```
NAME = SMITH, JOHN

SEX  = M
```

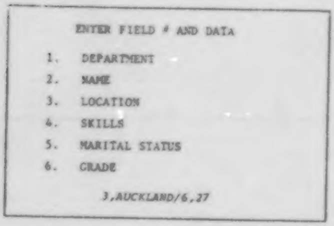
Fig 2D

Keywords

Used when -

a) Only a few of many possible inputs are to be used (Often the case with retrievals using weighted search arguments).

b) When it is desirable to enter input fields in any sequence.



```
      ENTER FIELD # AND DATA
  1.  DEPARTMENT
  2.  NAME
  3.  LOCATION
  4.  SKILLS
  5.  MARITAL STATUS
  6.  GRADE

          3,AUCKLAND/6,27
```

Fig 2E

Field-Identifier Directly Associated with & Entered with Data (Code-Link-Technique)

Note use of comma as field delimiter and slash for terminator of the data sub-group.

The same approach can be used with a light pen pointing to the dataname followed by keying the data.

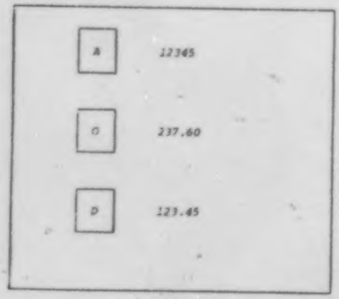
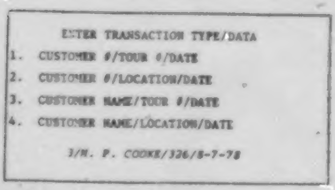


Fig 2F

Field Identification Provided with Special Keys (that also act as field delimiters)

A	Key Labelled ACCOUNT NUMBER
[illegible]	Key Labelled OPENING BALANCE
D	Key Labelled DEPOSIT

(Use of the special keys could precede or follow the data)



```
     ENTER TRANSACTION TYPE/DATA
1.  CUSTOMER #/TOUR #/DATE
2.  CUSTOMER #/LOCATION/DATE
3.  CUSTOMER NAME/TOUR #/DATE
4.  CUSTOMER NAME/LOCATION/DATE

       3/M. P. COOKE/326/8-7-78
```

Fig 2G

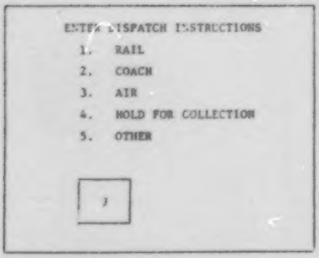
Field-Identification is with the Transaction Format Nominated with Data (Sequence-Key-Technique)

Data is positional

The contents of the first field is inspected and then the format for the remaining fields is known.

/ used as field delimiter

ACTUAL DATA ENTRY



ENTER DISPATCH INSTRUCTIONS

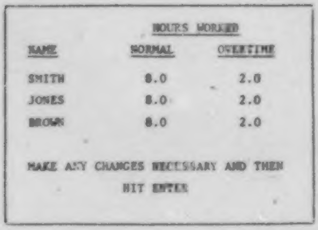
1. RAIL
2. COACH
3. AIR
4. HOLD FOR COLLECTION
5. OTHER

3

Fig 2H

Default - Entry (Single Field)

The most likely value (3) as shown, can be accepted (by pressing ENTER) or it can be changed and the new value taken.



	HOURS WORKED	
NAME	NORMAL	OVERTIME
SMITH	8.0	2.0
JONES	8.0	2.0
BROWN	8.0	2.0

MAKE ANY CHANGES NECESSARY AND THEN HIT ENTER

Fig 2I

Default - Entry (Form or Panel)

The most likely value for a number of fields is shown. These can be accepted or modified directly on the screen.

This approach is suitable for data entry situations where limited change is expected. It is also suitable for master file amendments as the new information can be keyed directly over the old.

the example from figure 2A should also be accepted when keyed as:-

123456 11/11/78 168.48

OR *123456* *11/11/78* *168/48*

Such tolerance makes for fewer input errors and makes the data easier to read while on the screen.

The variable length approach does require the depression of a key to denote the end of a data item. However it has advantages compared to fixed form:

- No forced counting of characters to fill fields
- Easier to read
- Field size readily expandable without operator retraining
- Usually fewer overall keystrokes since no leading zeros or trailing blanks are required
- Fewer input errors
- A more acceptable interface

When the value of a data item is not known this fact (ie. that it is not known) has to be indicated in some way.

A clumsy way is to repeat the delimiter (if not blank) giving for example:

123456,,168.48

Such a non natural expression is very error prone. A more natural way is to use (say) the query symbol to denote an unknown or unresolved value.
This would give:

123456, ? , 168.48

In some cases the user may wish to authorise the system to supply the missing information either from an earlier record or as some system held value or default.

For example keying:

*123456 , * , 168.48*

Where the * means take the appropriate date from the previous entry.

This usage can be extended to form a very convenient method for error correction by field replacement. For example to correct the second field you would key-

*Correction: * , 11/11/78*

3.1.4 Self Identifying Data

An alternative to deducing the identification of a data item from its position is to make or have the data self identifying. One widespread approach is the use of keywords as shown in figure 2C. Self identifying data permits the items to be entered in any sequence. This is particularly useful when only a few of many possible values have to be specified. Often the keyword can be omitted if from the form of the data it is possible to deduce its identification.

For example 10/3/78 might be sufficiently unique in format to be taken as a date without further qualification.

Part numbers might also be a certain format, such as 7-2134-63 and be identified accordingly.

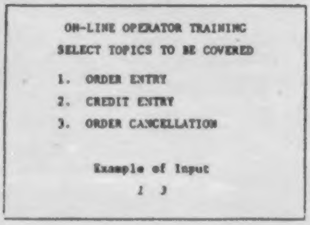
A variation is to combine the data delimiter with the data identifier by preceeding or terminating the item by special characters often generated from suitably labelled keys (See figure 2E).

3.1.5 Selection & Menus

Some portions of a dialogue will be operator initiated and the rest will be computer initiated. When the dialogue is computer initiated and if the situation is such that one (or more) choices have to made from a well defined, restricted list of alternatives a selection panel usually called a "MENU" is very effective (Figure 2H). The operator response is short, no coding is required and all alternatives are known. As additions and deletions are made the MENU can be changed. Often this is sufficient advice for the user and the problem of updating instruction manuals or operator retraining is avoided.

Menus usually require a single response (eg. Figure 2H) but multi-response Menus can be very effective in obtaining information used to limit the scope of subsequent questions and answers. (Figure 3A)

The example in figure 3B shows a Menu with three distinct and unrelated areas. Each column requires careful consideration as only one value from each column is acceptable. In this application, experiments showed that fewer errors resulted when three separate menus were presented to the clerical staff using the system.



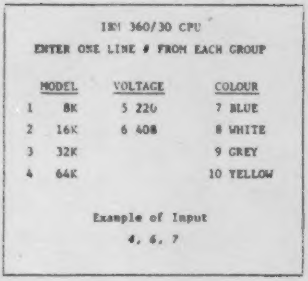
```
   ON-LINE OPERATOR TRAINING
  SELECT TOPICS TO BE COVERED

  1.  ORDER ENTRY
  2.  CREDIT ENTRY
  3.  ORDER CANCELLATION

       Example of Input
             1  3
```

Fig 3A

<u>Menu - Multiple Response</u>

Remarks

Suitable for casual user.
System may subsequently request more information as it systematically processes the options nominated.



```
              IBM 360/30 CPU
  ENTER ONE LINE # FROM EACH GROUP

   MODEL       VOLTAGE        COLOUR
1     8K        5 220         7 BLUE
2    16K        6 408         8 WHITE
3    32K                      9 GREY
4    64K                     10 YELLOW

            Example of Input
                4, 6, 7
```

Fig 38

Multi-part Menu

A multi-part menu requires multiple responses with at least one entry for each of several unrelated areas.

Remarks

Multi-part menus can be confusing. Tests have shown that frequently fewer errors result if separate single part menus are used each confined to a specific area of interest.

3.2 CAUSES OF ERRORS (PSYCHOLOGICAL FACTORS)

Studies (1) have shown that attention to key psychological factors can make the system much more usable and materially reduce the error rates. Care should be taken to avoid:

a) Operator "Channel overload": This occurs when the operator feels under pressure to respond quickly or to make a difficult decision. In some cases it is better to slow down the rate in which information is presented to the operator. A short time can be allowed to elapse between showing the outcome of the last action and presenting a complicated set of instructions relating to what has to take place next. Other techniques are to have only one idea for each display panel and/or to ask only one question at a time.

b) Operator Boredom: For the skilled operator who is very familiar with a given application it can be very tedious answering one at a time a series of well understood predictable questions. This is the opposite of the "channel overload" situation and a balance can often be found by providing the skilled operator with an alternative "fast-path".

(1) Reference 2. Chapter 17.

Such an approach allows the operator to anticipate future questions and nominate the answers in advance. For example if the operator knows that the sequence is NAME, STREET, TOWN, then the following could be entered in response to a prompt for the NAME.

A.B. SMITH,+12 MAIN STREET, AUCKLAND

Here the + symbol indicates the deliberate keying of the unsolicited input which if found to be valid causes the prompts for STREET & TOWN to be bypassed.

c) Operator frustration: An erratic response time for a given type of input can be very frustrating to the operator. A consistent response time is preferable to one that fluctuates between very short and intolerably long.

d) Operator uncertainty over the next action: We should avoid the situation where the user is sitting at the terminal in a quandary about what can be done next, how to do it, why the last response contained an error or was the last input even noticed by the system.

Under such conditions the confused operator is likely to make a hasty guess with all that that implies.

e) Reliance upon the Operator's short term memory: In most applications, several items of input are logically grouped together to form some unit such as an order, product specifications, a new policy etc. As items are entered it may be necessary to refer to earlier input as hard copy documentation may not exist (eg. telephone, order entry).

Therefore it is good practice to allow the related questions and responses to build up on the display. Since the number of entries can exceed the capacity of the display some form of forward and backwards scrolling becomes an automatic requirement.

(Also after an interruption the first action usually is to establish the last item processed.)

3.3. KEEPING THE OPERATOR INFORMED

3.3.1 Providing Feedback

Operating the only terminal at a remote location can be lonely and difficult, especially if you are not sure about what you are supposed to be doing. To a lesser extent, this is true at any terminal.

For the operator to feel comfortable, certain psychological and information needs have to be satisfied. These include information concerning :

- The Current Action

  It is re-assuring to know that the action you have taken (pressing a key, using a wand or light pen etc) has been recognised. This requires some form of immediate acknowledgement. Often this is the appearance on the display of the character keyed and a click or other sound to denote completion of the keyboard operation.

- The Maintenance of Environment

  It was found with the early time sharing systems that it was desirable to be able to confirm to the terminal operator that the connection had been maintained even if no output was yet ready. This was frequently done by sending uppercase/lowercase shift characters causing the type element to move and give out a recognisable sound. Now the logic within the terminals can be used to monitor the link status and activate suitable indicators.

- External Events

  The operator's job can be eased if some indication is given that some external action has taken place that will require some action at the terminal at a time convenient to the operator. For example, indicators could show that messages are waiting or that system shut-down is about to start.

The information to be presented should include:

- Application Status

  Which application is currently active?
  What stage has been reached?
  What mode of operation is in use?
  The outcome of the last transaction.

- Data Being Entered

  In many cases especially for casual users, prompts may be issued to solicit data and provide information on how the data is to be entered.

- Valid Next Options

  Where an application consists of a number of transactions that form a logical unit of work, some commands may not be permitted, nor may the unit of work be discarded before completion.
  The operator, therefore, needs to know which subset of commands is valid at any particular time.

- Device Status and Communication Link Status

  Confirmation that the hardware is working correctly.
  Status of the link to the HOST processor.

As far as practical, fixed positions on the display should be used for the operator information with the minimum of variation between applications.

3.3.2 Providing Prompts

A prompt provides the operator with guidance on how the next field (or group of fields) is to be entered. It may take the form of a field name (or back lighted text) and may provide additional information on field format, content, or meanings.

Prompts are particularly useful for the casual user.

When deciding if a prompt is needed consider :

- How much time is needed to transmit and display the prompt? (This was a concern with typewriter output devices). Now it is often possible to generate the prompt from a local controller and/or leave the prompt displayed on the screen.

- The display area needed. Is a meaningful prompt going to take up so much space that there is not enough room for logical arrangement of the input data? (Using a different display intensity helps distinguish between prompts and data).

- Can a full prompt or additional information be readily obtained if required.

  This is often the best solution if a long explanation is needed. A short prompt can be shown along with a symbol to indicate that more information is available on request.

- Could the operator be provided with the information any other way? On-line information solves many of the problems of updating instruction manuals at branch offices and getting people to read them!

A prompt is usually displayed once it has been determined what the user is attempting to do, ie. the prompt is the initial response once the prime transaction-type is nominated.

EXAMPLE

REMARKS

```
           ENTER  QUANTITY

                 OR

             ENTER  QTY
```

Fig 4A

Prompts where the field name (or abbreviation), only is given, it being assumed that the user knows the field attributes.

```
      ENTER QUANTITY (Numeric)

              - - - - -

    ENTER QUANTITY        xxx.xx
    ENTER QUANTITY          3.2
```

Fig 4B

Prompts for numeric fields.

```
  CUSTOMER # ..........................

  NAME       ..........................

  TOWN       ..........................
```

Fig 4C

Prompts with each data element commencing at a fixed screen position but with variable length data (maximum field size shown). (This format is often called "Adjacent Form")

```
  ............................................ NAME
    LAST            FIRST           MIDDLE


  ........................................ PERMANENT
  ........................................   ADDRESS
  ........................................
    SUBURB                 CITY


  ............     ...............       TELEPHONE
    HOME             BUSINESS              NUMBER
```

Fig 4D

Here the intent is to reproduce a "manual" form as closely as possible. Blank lines between areas also aid clarity.

For example, in a banking application, pressing a key labelled 'DEPOSIT' might generate the prompt below :

ACCOUNT NUMBER, CASH AMOUNT, CHEQUE AMOUNT

Other examples are given in Figures 4A-D, 2A-C, 2E & 2G-I.

3.3.3 Providing On-line Information

As the number of terminal sites increase the problems of providing "detailed pertinent current" information on how to use the system multiply. These are compounded if an ever changing population of users use the system.

Attempting to distribute printed documents in the form of instruction manuals is doomed to failure. They are either out of date, have pages in the wrong place, are missing, or are stored at a location other than where that user currently has a problem.

Some instructions have to be provided no matter how good and natural the interface. As on-line storage costs decrease it makes good sense to put this information on-line and have it directly available at the terminals.

Two methods of retrieving the information should be provided :

a) By request: If an operator wishes to study the instruction a BROWSE-type capability is needed.

b) In Context: Usually the information sought relates to a very detailed point arising during the entry of a transaction. The information sought is about the permitted values, the format of a given field, the syntax of a command etc. The need for such information can be signalled by locating the cursor at the field in question and pressing a key labelled HELP.

This is easy to use and can apply to all inputs. If no on-line instructions are held for the particular query this can be advised.

Pressing an ENTER key after reading the page(s) of information would revert the system back to its previous state.

Such an approach allows the on-line operator information files to be progressively built up catering for known problem areas as these arise with the most common being addressed first.

Once set up, the facility can be very useful as an application expands to include some new area over which confusion can be expected in the initial stages as the operator can resolve the problem at the terminal.

3.4 TOLERANCE

In numerous ways the rigidity of an input method can be eased with no loss of accuracy and significant gains for the user. Some of these are discussed below :-

3.4.1 Provide Natural Exits

Experience with Medical systems, Computer Assisted Instruction and other applications has shown that a user must have the opportunity to "opt-out" if the going gets rough or if some other application is required. This is best done by a command key that causes a menu to be displayed that provides for graceful termination of the current activity.

3.4.2 Abbreviations

An irritating feature of many systems is the lack of tolerance to minor mis-spelling in keywords or command names. A good practice is to permit abbreviations and alternatives. Such tolerances should be withheld for commands such as ERASE where accidental use could be serious.

eg. Keywords :

Name = Smith or *N = Smith*

(Date of Birth) *BIRTH = 10/2/40, DOB = 10/2/40*

Commands	*PRINT*	*or*	*PRT*	*or*	*P*
(Alias)	*LIST*		*LST*		*L*

3.4.3 Substructure

Some data has a substructure associated with it. Any dialogue format that forces the operator to deviate from this natural form will inevitably cause errors. In New Zealand, for example numerically expressed dates are written in the form DD/MM/YY. (In some countries the form is MM/DD/YY). It would be looking for trouble if a date was required to be entered in the form YY/DD/MM.

For most people their names are expressed with the surname last. Yet we are constantly required to write the surname first even on documents only used for computer entry.

3.4.4 Repetition

People normally document variations to data rather than repeat the actual values. Figure 5 shows such an example. The dialogue should accommodate this by permitting earlier entries to be carried forward. (Perhaps using the " or * symbol to mean ditto.) Each subsequent display will force the correct column alignment to aid sight verification.

In a different context (See Figure 6) some applications require the entry of complex specifications for the ordering of a product. Often successive products are only minor variations upon the first. (eg. A number of VDU's forming a cluster may differ only in the keyboard layouts) It is (the author assures you!) very frustrating if the user is forced to answer each question again and again. A far more user-oriented approach would be to use the first answers as suggested defaults and so let the user concentrate only on the variations between products.

QUANTITY	CATALOGUE REFERENCE	DESCRIPTION	SIZE	COLOUR
4	213	Exterior Paint	4L	Red
7	506	" "	"	Blue
6	129	" "	"	Brown
3	247	Interior Paint	"	"
8	914	" "	2L	"

FIGURE 5 SAMPLE ORDER

3.4.5 Decimals

Some numeric fields may only have integer values. Other fields may involve decimals. In each case entries such as 2, 2.0, 2.37, should be able to be keyed and the editing detect any invalid entries. This is easier for a casual user than to have rules that require field A to be integer, field B to be an integer and two decimal positions but without the decimal point (such as a money value expressed cents) and field C to be keyed with a decimal point (2.37).

3.4.6 Better Editing

As an industry we can do a better job of editing input data. Too many of our systems reflect an almost pathological obsession that says we must minimise the use of computer cycles and storage. Yet today hardware costs are decreasing and people related costs are increasing. Range testing is a good example. It is fine provided your goal is just to restrict the value to within some prescribed range.

In practice, however, the upper limit may (must) cater for the largest value expected. When systems are set up this may not be known and a best guess is used.

Glib and Weinberg (1) discuss Adaptive Checking. This approach utilises the computer logic and storage to "remember" past experiences and patterns. Using this accumulated experience the limit tests are based upon recent actual observed cases.

When a value outside the range is encountered the system asks the operator to confirm that the value entered is correct and asks if the value be recorded as the new upper limit. (Obvious large one-time orders would thus not cause the upper limit to change).

Many designers hate the thought that anyone but Senior Management or System designers should set limits. Yet with a sensible compromise that reports any large change to a limit on an exception report a self adapting system is very viable. If an automatic downwards adjustment of say 10% is applied each month even a poor judgement to change a limit will have little long term effect and quickly would be more effective than the somewhat arbitrary limits in use today.

Use of Patterns

A human looking at data often has sufficient understanding of the real life situation to recognise probable anomalies in input values. Suppose that in a warehouse where items are despatched it was found that for items normally ordered in multiples of one dozen that the only variations experienced were for an additional half dozen or when very few items were needed the range was for 1 to 3 single units. Knowledge of such patterns can form the basis of better editing than simple range tests.

(1) Reference 1. Chapter VII.

Making sure that questions are understood

To ensure that the user has correctly interpreted questions and/or given the data in the required form it is good practice to display back on the screen the meaning(s) interpreted by the system. This can either be after a response has been given or when several responses relate to each other a summary at the end.

3.5 ERRORS AND THEIR CORRECTION

3.5.1 How Much Data Should be entered at one time

An obviously critical decision concerns the number of data fields to be keyed before program controlled editing commences.

When the information has for the user some natural sub-groupings then these form a good basis for grouping the data entry (eg. an invoice line item).

If however, the dialogue is computer initiated based in part on earlier input and where the user needs to ponder over what to enter, then each item is best entered by itself.

The basic choices are between :

Field By Field	:	(Also known as Question & Answer Mode) Each field is keyed and then edited before the next field is keyed.
Logical Grouping of Fields	:	(Records or sub-records). Each field is keyed using delimiters or positioning and then all fields are edited before the next group is keyed.
The Contents of an Entire Screen	:	(Could be many records). Many fields are keyed and then edited before the next full screen is keyed.

In making a choice, the designer must consider :

- Ease of error correction.
- Ability to provide an adequate response time.
- Suitability for application and operator skill level.
- Amount of editing to be performed.
- The editing capability of the terminal and its associated controller.

EDITING : Two levels of editing can apply:

a) Field Editing In Isolation

Tests can be made to ensure that each field has contents that are in accordance with the characteristics defined for that field :

eg. Numeric with a Sign,
A Self Checking Number, within the Range 900 to 10,000 etc..etc

b) Field Editing in conjunction with other fields and Data (Relationship tests)

Tests can also be carried out to ensure that the valid relationships are maintained between the various fields entered and other applicable data :

eg. If a SHIP-TO-NAME is entered, then a SHIP-TO-ADDRESS must also be entered.

Often the decision of which tests to apply is determined in part by the contents of the fields being entered. Situations can occur where most of the data is correct, but that the wrong tests were carried out. When this happens the resulting "error messages" can be very confusing.

All of which leads to the conclusion that advising the operator and the actual correction of the apparent errors detected during on-line editing can be quite complex.

3.5.2 Corrections

Backout: An ability to cancel earlier entries and back track is needed. This arises if the operator detects a mistake with an earlier entry that has already been accepted.

Example 1: In an order entry situation if stock inventory is being updated then any such entries will have to be reversed. e.g. 20 of product 1234 ordered instead of 20 of product 1243

Example 2: Consider a Question & Answer order-entry application capturing product specifications. (Figure 6)

Q1 WHICH PRODUCT GROUP?

A ......*FREEZER*..............................

Q2 CAPACITY IN CUBIC FEET?

A ......*12*..............................

Q3 CHEST TYPE OR VERTICAL TYPE?

A ......*VERTICAL*..............................

Q4 DOOR HINGES ON LEFT OR RIGHT?

A ......*?*..............................

Fig 6 Entry of Product Specifications

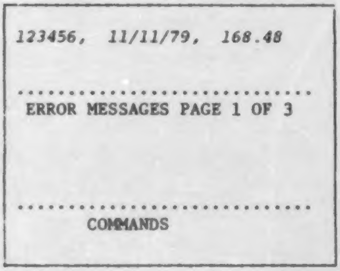
Here the response to Q1 obviously generated the Prompt for Q2 and so on. Suppose that at Question 4 the operator realises that the order is actually for a chest type freezer in which case Question 4 would not apply and Question 3 must be re-answered.

The operator needs the capability to effectively return to Question 3 and recommence from that point. This might be implemented as a general re-start from a fixed point working forward again offering each earlier response as a default.

Advising Errors

Providing sufficient space on the display is not usually a problem for the text to describe errors relating to a single field. When multiple fields are entered together and several errors are found and/or relationship errors are found it may be necessary to use most of the display space for explanatory text. Once the error is corrected then the previous display (series of related entries such as an order) can be restored with the new transaction added.

If necessary several error-text displays might be needed and a split screen approach can be useful. See figure 7.



```
123456,  11/11/79,  168.48

..............................
 ERROR MESSAGES PAGE 1 OF 3



..............................
       COMMANDS
```

Potentially applicable Data on display with error fields in high intensity.

ERROR TEXT(S)

Details of how to page back/ forward through the error texts and how to return to the main display.

Figure 7

3.5.3 Entering the Correction

The method of entering the correction will depend upon the screen layout.

When layouts such as figure 2C are used the new values can be keyed directly over the old values. When free form input is used (Figure 2A) then methods such as that discussed in section 3.3.1 are applicable or the entire entry re-keyed.

A different problem arises when the course of editing can be modified as discussed for figure 6.

For each of these cases the old input modified by corrections is submitted to the edit tests in the same way as new input. This simplifies programming and makes for a more consistent handling of errors.

3.6 TESTING THE PROPOSED DIALOGUE

Any proposed dialogue should be evaluated as fully as possible prior to final implementation. This can best be done with actual screen displays even if no processing takes place. Keying directly onto a screen lets the future user study the layout and allows for machine printed documentation for the analyst. For key projects trials with more elaborate simulated interfaces are worthwhile.

Simplicity with the dialogue should be the goal. This will mean that most transactions can be handled in a straightforward way with the complexity only being introduced and becoming apparent for the more unusual cases.

Care and attention at this stage reduces the need for explanatory documentation. Excessive documentation is an admission of failure and is too much like constructing a new road and installing a road sign saying *"Uneven and dangerous surface"*.

3.7 PRINCIPLES FOR DIALOGUE DESIGN

The following are key aspects of Dialogue Design:

1. Display only a little information at a time
   A very crowded display is inviting the user to skip over portions.

2. Keep to a single theme
   Comparisons have shown that faster clean input is achieved with only one idea on the display at one time.

3. Tidy up the screen
   As each logical unit of information is ended the screen should be cleaned up. (eg. at the end of an order) If interim instructions are shown to aid error correction these should be removed as soon as possible so that the display is either clear or shows only the net valid inputs.

4. Have short operator responses
   Short responses are more acceptable to the operator, especially when under pressure or engaged in other activities such as talking to a client in a telephone sales order entry situation.

5. Never leave the operator in doubt
   No repeat no input should go without some acknowledgement. This might be the unlocking of a keyboard, sounding an alarm or output on the display. At all times the operator should get a response. This cannot be stressed too much. An operator relaxes once aware that his/her actions have been completed correctly. (ie. Responsibility for the next action rests with the system) Therefore whenever a long delay is expected, such as when a complex file retrieval must take place, then some interim acknowledgement to signify receipt of input from the keyboard should be given.

6. Strive for clarity
   Make the display as easy to read as possible. Avoid overcrowding and make use of special characters such as dashes to divide up the display. Have the questions and responses in different intensities or use upper/lower case.

7. Strive for Similarity
   Within a location each application should be as consistent in operational approach as possible. This applies to layouts and the use of special keys (eg. delimiters, commands and menu options).

8. Avoid Potential Confusion
   Make sure that people do not have to stop and puzzle out what is meant. For example less time is needed to understand ;

   "NOT KNOWN" than "UNKNOWN"
   "IS NOT" than "ISN'T"

   It is therefore best to test dialogues with the potential user group so that instructions are clear to them.

9. Make it easy to request help
   At any stage it should be possible to get help or to terminate the current processing.

4.0 CONCLUSION

People's satisfaction and capability to perform arises in many ways. Efforts to increase "efficiency" in one area by reducing the variation in tasks or speeding up individual work processes may be sub-optimisation when the wider picture is examined. For example if clerical coding of documents is used to reduce the subsequent number of keystrokes or if work tasks are so simplified to reduce training that the lack of variety makes people dissatisfied and careless, any "gains" in the DP area may be more than offset in adjacent areas.

Many computer applications are viewed only as technical problems to be solved with technology by technicians. The failure to recognise, understand and take into account human issues can be disasterous, affecting not only costs, efficiency and staff morale but in extreme cases a company's ability to operate. People are most adaptable with an ever accumulating knowledge of their work environment. A good systems design capitalises upon these strengths and the knowledge so gained by making the system dialogue as natural as possible and by letting the user exercise judgements in those situations where she or he has special knowledge.

At all times the systems designer must understand and remain aware of the business objectives and needs of the user. The emphasis is changing and it is no longer satisfactory *just to get something to go*. We must do better.

The computer system should not be a rigid soul destroying straight jacket that is pedantic to the extreme over input formats and terse to the point of rudeness in its condemnation of our apparent stupidity. Economically viable, ergonomically sound systems featuring tolerant interfaces that are "People-oriented" are required.

We, the system implementers must ensure that such expectations of our systems are realised.

---oooOOOooo---

5.0 REFERENCES

1. "Humanized Input - Techniques for Reliable Keyed Input". Tom Gilb and Gerald M. Weinberg. Winthrop Publishers 1977.

2. "Design of Man-Computer Dialogues". James Martin Prentice Hall. 1973.

3. "Chatting with Computers". C.R. Evans. IEE Electronics and Power. April 1975.

4. "Interactive Human Communication". Alphonse Chapanis Scientific American. March 1975.

5. "Improvements in Man/Machine Interfacing". EDP Analyser. April 1975.

6. "CRT Displays for Power Plants". M.M. Danchak. Instrument Technology. October 1976.

7. "IBM 3270 Screen Design". IBM Form SR20-4441 January 1975.

8. "Computers for People-Designing Human Systems". Margaret Exley and Neil Harding. Data Systems. February 1977.

9. "Response Time in Man-Computer Conversational Transactions". Robert Miller. AFIPS Conference Proceedings 1968.

10. "Humanising Data Entry by Default". Tom Gilb and Gerald Weinberg. Datamation. August 1976.

COMPUTERS, COMMUNICATIONS AND THE FIRE SERVICE

AUTHOR R.E.M. COOPER, B.Sc. (Hons.), Ph.D.
Senior Lecturer in Computer Science,
University of Canterbury.
Christchurch.

ABSTRACT

The procedures associated with maintaining adequate fire cover during a major fire incident or multi-fire situation has been a major problem to be undertaken by fire brigades. While the centralisation of facilities for recording the despatch and arrival of fire appliances has been of assistance, the monitoring of fire cover remains a manual process which is entirely dependant on the experience of the mobilising officer to cope with an individual situation.

In order to investigate the feasibility of using computers for this purpose and in the wider sphere of computer-based mobilising, a pilot project has been undertaken to implement an on-line fire appliance mobilising system.

KEYWORDS AND PHRASES

Automatic sending of messages, micro-processor, communications, alarm, operator communication, response.

C.R. CATEGORIES

2.11, 2.12, 3.29, 3.81, 3.83, 3.89.

1. INTRODUCTION

This paper outlines a small experimental project that was set up to investigate the use of computer equipment to support the emergency operations of the Fire Service. While extensive work has been carried out overseas to investigate how computers can be utilised directly in linking communications between fire stations, the concentration here was initially on aspects of communication between the operator and the machine. Thus there is a requirement for rapid access to operational data, and the display of data in a clear manner, to assist with provision of adequate fire protection and fire fighting in a given area. Some of the benefits that follow include the automatic production of management reports, the monitoring of adequate fire "cover" and the flexibility to cope with growth caused by centralisation.

2. FIRE SERVICE REQUIREMENTS

The aims of the Fire Service are primarily to provide a fire fighting service to the community, although the lesser ones of fire prevention and training have some impact on the design of any system. This service is centred around a number of local fire stations distributed throughout a population area so that there are fire fighting facilities available corresponding to some measure of the fire risk. Until recently, when a fire occurred, the local station was notified by some means, and the necessary fire appliances were mobilised, with a fire station further afield being contacted by telephone or radio if the need for further appliances arose. In a suburban area, this could result in two nearby fire stations being turned out for the same fire without either of them being aware of it.

Nowadays, the centralisation of communication services, and in particular the telephone, has forced drastic modifications to the control of fire services. The most common sources of fire alarms are through emergency telephone (111) calls and private fire alarms, the latter being transmitted by communication links from a building to the fire station. Emergency telephone calls are routed through a central operator to the appropriate service for action, which while being more efficient, with respect to manpower, can generate anomalies. For example, an emergency telephone call in Methven, fifty miles from Christchurch will eventually arrive in the watch-room at the Christchurch Fire Station where the mobilising officer may telephone the local volunteer brigade in Methven, to turn out to a fire, only a short distance from where the original call was made.

Within a large population area, there is now a centralisation of fire control in usually one or two watch rooms. These are manned by operators on a 24-hour basis whose activities include:-

(a) The recording of details about any fire call that arrives.

(b) The identification of the location of the call and the resources that should be mobilised to answer the call. These are usually obtained by reference to appropriate files.

(c) Activation of any other back-up services that may be required.

(d) Recording details of the progress of an incident as they are radio-ed in, and carrying out any subsequent mobilising actions that may be required.

(e) Ensuring that all sections of the population area under surveillance has adequate fire cover at all times, and, if necessary, moving appliances between stations.

(f) The production of a hard-copy report, which may be analysed for statistical and planning purposes, recording the actions that have taken place.

While a clear statement of the actions that take place is possible, the pressures on the operators are considerable, particularly when there are a number of incidents current, and there are extensive inter-station movements underway. Furthermore, since most operations take place under emergency situations where time is critical, rapid access to data is important.

3. COMPUTER SOLUTIONS

A number of computer systems have been developed overseas to facilitate operations within the fire service. These have generally been established to attempt to solve particular problems and their emphases consequently differ. It is probable that an all-embracing computer system should include some or all of the components shown in Figure 1. When an alarm occurs, it may either arise from a signal transmitted from an automatic alarm and input directly to the processor, or indirectly through an operator-entered transaction on a key-board. On initiation of a call, a new incident record is created and appropriate details of appliances to be mobilised and routes to take are indexed. When a more extensive communication system is included, mobilising and informative messages are automatically transmitted by communication lines or radio to printing devices in the fire appliances or fire stations. Any other authority, such as police or traffic control can also be notified directly.

During the course of an incident, the status of the operations are continually being updated by data entered through keyboards. These may be further mobilising commands, or informative or request messages from the senior officer at the fire ground sent by radio to the control room. Since these messages will often be coded both for reason of security and brevity, the messages are decoded and the actions are taken. For backup and record purposes, all transactions are date-stamped and logged.

Larger files with low-activity may be established for the purpose of long-term storage of attendance details, or incidents of special concern, as well as providing access to data necessary for specific emergencies. Thus, chemical hazards and special risks are stored so that they can be readily accessed.

The following are a number of locations overseas where there have been major projects developed:-

(a) Glasgow. Here a computer system has been used for a number of years with the emphasis on maintaining the files of buildings, their plans and locations of detectors, alarm boxes and the like. When an alarm occurs in a building known to the system, the relevant data is extracted and can be transmitted by radio to a facsimile printer in the cab of any fire appliance. Thus the details of the building can be made available while the appliance is en route to the fire ground, without any inherent delays necessary in producing them.

However, this has been generally less than successful for a number of reasons. The cost of installing the radio transmission and receiving equipment to enable the plans to be transmitted is high and the printing devices themselves are very slow. Furthermore, the advent of relatively cheap microfiche facilities means that it is possible to extract and print the details of a building within the watch room while the call details are being recorded and these can be taken out to the fire by the fire appliance or in a support vehicle.

(b) Los Angeles. [1]. This City Fire Department operates over 110 stations spread over an area of 400 square miles. Obviously, the problems of co-ordinating mobilisation and providing effective cover were major factors in deciding on the use of a computer system some years ago. The result is a fully integrated approach utilising ten dispatcher consoles, two supervisor consoles, up to ten back-up consoles and a comprehensive set of display boards. Into each of the consoles is built a VDU screen and key-board, a telephone control panel, microfilm screen and key-board, short-term cassette tape recorder, radio control panels, and digital clock. The operator thus has ready access to a wide range of sources of information to assist with the operations.

From a technical point of view, however, this also is not a particularly successful system. For reasons which are not immediately apparent, the computer resources are insufficient to maintain a satisfactory response rate under reasonable load conditions. Consequently, operator interaction is minimised and commands are issued using relatively complex "command strings". Whereas in many areas the use of these can result in efficiency of processing, it is considered that from the human engineering viewpoint, this is not satisfactory in an environment where time is critical, and the likelihood of making a mistake should be minimised.

Furthermore, it is clear that when input errors occur, the assistance given to the operator is often unsatisfactory. Thus if a command to initiate a new fire call might be entered as

"I/T=SF/A=FIGG,X,NEUTON/P=967-8112/C=TAC2/"

to which the reply received is

"NO HIT ON INCIDENT LOCATION"

whereas a list of names which either look or sound similar to FIGG and NEUTON, would assist considerably.

(c) Brooklyn. [2]. The borough of Brooklyn is currently operating a Management Information and Control System (MICS) for supporting its fire appliance mobilising operations. This is a pilot project which will eventually be integrated into a comprehensive system for the whole of the city of New York, supporting 350 pieces of apparatus, 10,000 firemen and answering over 450 000 calls each year. Currently, the system is based on dual PDP-11/45 computers with three large-capacity disc drives and up to 16 VDU terminals. In addition, there are a 100 specially-designed teleprinters with response panels located in the fire stations throughout the borough.

Alarms are received either through old mechanical street alarm boxes, newer electronic street boxes that permit voice communication as well as transmission of digital signals, and by direct telephone contact. The computer determines a suitable mobilising response for each alarm, displays this for approval or modification by the dispatcher, and transmits the alarm data to the corresponding fire stations.

Status is maintained by a micro-processor-based display of both the current incidents and the fire appliances. In the case of a failure of the central processing system, the microprocessors can assume full control of the status recording subsystem.

This is a comprehensive fire mobilising and recording system designed for a large-scale fire service. It uses up-to-date technology in the form of micro-processors, and provides extensive fall-back capabilities. However, it is not clear whether these represent a degree of "over-kill", which might be overcome by the use of manual back-up. Obviously, it would be extremely difficult to justify the cost of the full system even in the larger New Zealand fire control rooms which handle a number of incidents two orders of magnitude fewer, whereas one with reduced resources but the same aims might be.

4. DEVELOPMENT OF THE CHRISTCHURCH SYSTEM

Interest in developing a project involving the use of computers for the Fire Service began in 1976 with a request from the Christchurch Fire Brigade to the Department of Computer Science at the University of Canterbury for assistance. The project was adopted and used as a major teaching assignment during 1977 when a class of students, working on it in small groups, produced a number of suitable designs. Subsequently, the Fire Service Commission provided a Research Grant which was used to employ two students as programmers during the summer vacation, during which time, the majority of the development work was carried out.

The particular design objectives which were discussed included the following factors:-

(a) Centralisation of fire control within the Christchurch metropolitan area had already been achieved and all emergency telephone calls were directed to the central watch room. Furthermore, all private fire alarms transmitted their signals here so that there were no potential problems with distributed private fire alarms.

(b) The storage of relevant data for automatic alarms was of low priority because a microfiche facility had already been established. This enabled rapid access to both the details of the fire location and the units that should be mobilised, as well as displaying floor plans of the building, locations of entrances and so on. However, because this data at some stage had to be entered into the computer system, a minimum file of data was established.

(c) Operationally, the system would be used by the watch-room staff who, while having some experience with the use of typewriters, are not extremely skilled in this area. Consequently, the man/machine communication aspect had to be designed carefully so as to be relatively simple to use. Experience with time will indicate whether more complex command strings can be employed either instead of, or as an alternative to, the present arrangement.

(d) As this was an investigative project only, there was no attempt at incorporating automatic sending of messages or signals from the computer to outlying fire stations or fire appliances. It is envisaged that in any long-term solution this could best be solved by the use of one or more microprocessors communicating with the central computer.

Thus it can be seen that the system to be developed was primarily designed to test interaction with the operators and to establish its viability as a means of recording mobilising operations. Future expansion will readily allow for the incorporation of functions such as the automatic sending and receiving of messages.

5. IMPLEMENTATION

The demands of the fire service are such that 24-hour access to any computer system is necessary. This placed considerable restraints on the resources that were available to us. However, permission was obtained from the Christchurch City Council to make use of their PDP-11/45 for the duration of the investigation. This computer, which operates on a 24-hour basis, has as its main function the co-ordination and recording of traffic flow around the city and also supports a number of research, development and management functions for the City Council. It currently runs under the RSX-11D operating system.

While it is possible to consider the project as just another on-line transaction processing system, it has a number of aspects of human engineering which make it different. These usually result from the environment in which it is operating, and occasionally from other constraints.

(a) In a conventional transaction processing system, the operator initiates a transaction, enters the required data, with corrections if editing show these to be necessary, and on concluding starts on the next transaction. Here, there is a need for a priority structuring of processing since the arrival of an alarm must be able to be processed immediately by the operator. That is, even if he is in the middle of completing an incident report, the alarm must over-ride that operation. This was catered for by having all input queued through a control procedure which checked for a unique combination of data indicating an alarm call. While this means that the alarm is not processed until the next read request is initiated, in practice this has not proved to cause any significant delay.

(b) The DEC VT52 terminal, on which the development work was carried out, has a numeric key-board which can be operated as function keys. These greatly facilitated the initiation of processing modules, which include:-

Alarm module
Street index
Status display
Private alarm index
Incidental data logging
Report generation.

(c) Data display. There are two main sources of data to be displayed if the control room operator is to be aware of the overall fire cover at all times. These are the status of appliances at their respective stations and the status of any current fire incidents. It became obvious very early in the design stage that the optimum way to display the data would be through the use of two screens, one for each of the data sources. This corresponds to the type of displays employed by a number of large computers for operator communication where one screen will contain the sources of data, such as the input queues, and on the other the processing which is currently being carried out.

However, the project was restricted to one screen only and it thus has been necessary to restrict the displayed data to a minimum to enable it all to fit on one 24 lines by 80 character screen. This has resulted in a less than optimum solution, but one that was considered essential. An example of the display produced during the course of an incident is shown in Figure 2.

(d) Operator communication. As was mentioned previously, the basic requirement for data input was considered to be simplicity of operation. Thus, except where names of streets or specific data are required, data entry consists of menu selection requiring minimum of operator intervention. This creates heavier demands on the system since it increases the number of accesses required, but with the satisfactory responses obtained, it was considered to be a requirement. Experience gained over an extended period of time will no doubt suggest whether a simple form of command string would in practice be as suitable or even better than this.

6. CONCLUSIONS

While it is premature to reach any general conclusions about the success or otherwise of the project, the reaction so far obtained is of general satisfaction, particularly with the man-machine communication side. This was considered to be one of the most important aspects of the project.

Extensions to the scope of the project which are either being considered or are underway include:-

(a) Maintenance of a man availability file for call-back in the case of major incidents.

(b) Local statistics generation.

(c) Generation of a long-term file of incidents on magnetic tape for centralised processing. This will be used for management purposes to produce statistics, display trends in the types, locations and number of incidents and the requirements for equipment and training.

It is obvious that what is a small initial project will be the forerunner of developments in the larger metropolitan areas which will utilise a number of terminals and inter-connected microprocessors to provide a comprehensive man-machine communication system for the Fire Service.

7. ACKNOWLEDGEMENTS

Sincere thanks are due to a number of sources, without which this project would never have been successful. The Christchurch City Council City Engineer's Department provided free access to their traffic control computer, and very considerable support from Graeme Cox and his staff. Digital Equipment made available on loan a VT52 terminal on which to test out all the ideas and the Municipal Electricity Department installed the temporary communication lines. The programming efforts of David Anderson and Neil Olliver were supported by a research grant from the Fire Service Commission. Finally, the officers and men of the Christchurch Fire Brigade, especially Deputy Chief Officer G. Roberts and Station Officer K. Wayman, without whose time and support the project would not have been possible.

8. REFERENCES

[1] Los Angeles Fire Department Command and Control System Training and Operations Manual. Vols.I and II.

[2] Computer Design. pp48-68, November 1977.

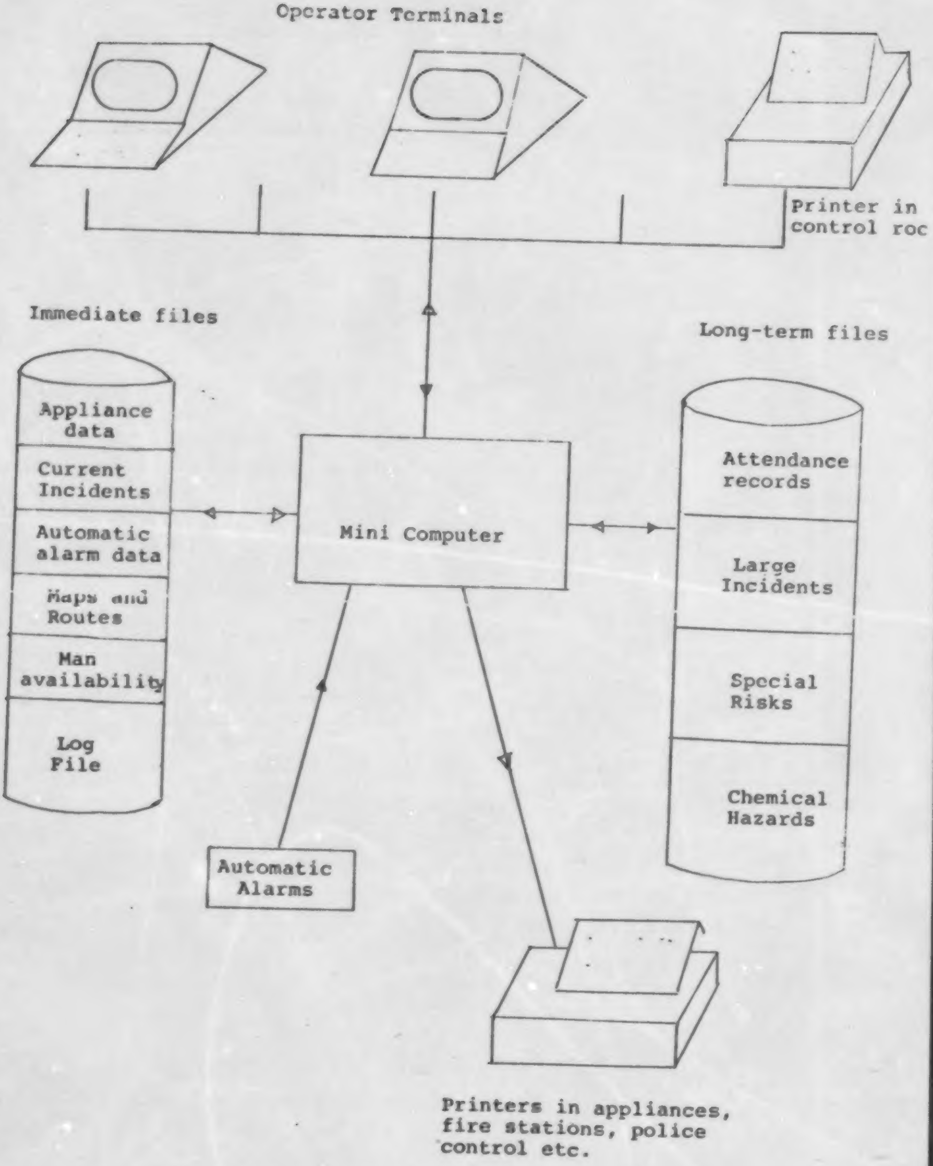
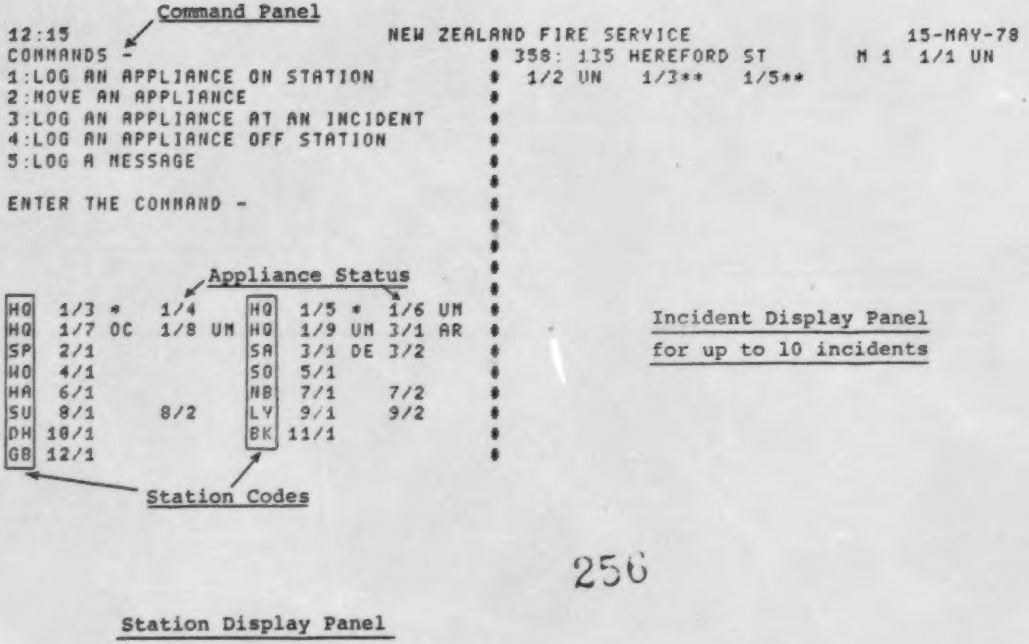

Operator Terminals
Printer in control roc
Immediate files
Long-term files
Appliance data
Current Incidents
Automatic alarm data
Maps and Routes
Man availability
Log File
Mini Computer
Attendance records
Large Incidents
Special Risks
Chemical Hazards
Automatic Alarms
Printers in appliances, fire stations, police control etc.

FIGURE 1.
Operation of a fire service mobilisation and control system



```
12:15                           NEW ZEALAND FIRE SERVICE                  15-MAY-78
COMMANDS -                                    # 358: 135 HEREFORD ST       M 1  1/1 UN
1:LOG AN APPLIANCE ON STATION                 #  1/2 UN    1/3**    1/5**
2:MOVE AN APPLIANCE                           #
3:LOG AN APPLIANCE AT AN INCIDENT             #
4:LOG AN APPLIANCE OFF STATION                #
5:LOG A MESSAGE                               #
                                              #
ENTER THE COMMAND -                           #
                                              #
                                              #
                                              #
                                              #
                                              #
HQ  1/3 *   1/4      HQ  1/5 *  1/6 UM        #
HQ  1/7 OC  1/8 UM   HQ  1/9 UM 3/1 AR        #
SP  2/1              SA  3/1 DE 3/2           #
WO  4/1              SO  5/1                  #
HA  6/1              NB  7/1    7/2           #
SU  8/1     8/2      LY  9/1    9/2           #
DH 10/1              BK 11/1                  #
GB 12/1                                       #
```

Command Panel
Appliance Status
Incident Display Panel
for up to 10 incidents
Station Codes
Station Display Panel

FIGURE 2.
Typical Status Display

OBSERVATIONS ON USER/VENDOR COMMUNICATIONS

AUTHOR: BRUCE R. DONALD,

Burroughs Ltd,
Wellington

ABSTRACT:

All computer installations include some form of user/vendor relationship, which if well maintained and I might add, well understood, can make a significant contribution to the mutual success of both parties.

Although both parties are seen to have a common goal, the very nature of the relationship and the separate self interests that must be served can be an influence towards eroding the relationship. It can be identified as often present in situations commonly interpreted as failures.

The best relationships are fostered where good communications exist.

This paper seeks to identify the attitudes and elements which can obstruct the achievement of the common goals, suggests ways they can be avoided, by presenting a perspective, on the common situations which give rise to misconceptions, with the ultimate objective of ensuring misunderstandings are not construed or develop into misrepresentation.

KEY WORDS AND PHRASES:
Communication, User, Vendor, Language, phrases, terms, contracts, User groups, N.Z. Computer Society, competition.

C.R. CATEGORIES:

2.9

Having at least incorporated the conference theme in the title it might be questioned whether a paper on this subject could appropriately be classified within the specifications set out in the call for papers.

If it posed a problem for the convenor it would be minor compared to the magnitude of those experienced by the author in development of the theme.

The first to fit the first criteria "as embracing communication between:-

- DP professions/industry and the public at large;
- DP professionals and computer systems;
- Communications between computer systems."

This conjured up the possibility of some user/vendor relationship experiences, which on occasions, given the appropriate circumstances could suggest classification into some or even all categories.

A further complication also arose in the choice of user/vendor phrase sequence particularly as it related to the "DP profession/industry" heading - which side does the Vendor fit or is that creature somewhat more considered the public at large?

Mental struggle to find some adequate solution finally abated and the thoughts dismissed on the grounds that this paper would be oriented to the very simple category of people communication and really had nothing to do with the "bits and bites" of the real computer world.

The second obstacle noted related to the class of paper.

While readily rejecting that it would have anything of a technical content and certainly not of 40 minutes hot interest, it seemed the structure and treatment should adequately suit a general paper of 20 minutes but avoid such material that could give rise to a full "workshop session with demonstrations!"

An unnecessary preamble? Or a subtle suggestion we could on occasions consider this very special relationship can have its tenuous moments?

In spite of the possibility of differing opinions on this point, it should be universally agreed that the user/vendor relationship commonly found in most computer installations can, if well maintained, make a significant contribution to the success of the enterprise.

Both user and vendor see this success as a common goal and yet there are many facets of the relationship which sometimes militate against this, not the least these being the separate self-interests that support their individual existence.

This at times can foster an adversary relationship which results in no satisfaction to anyone least of all the people involved.

It therefore seems that this mutual success is only possible if good communications exist.

What are some of the situations in which this secret weapon can be employed? There is no recognized priority of importance inferred in the sequence of the examples nor hopefully do they contain anything but an objective treatment.

Conscious that some of the material could fall into the category of "teaching a grandmother to suck eggs", priority perhaps should be given to the risk of not taking full advantage of repetition. All of us are guilty at some time of putting too high a value on our knowledge particularly when enduring again a presentation or pitch which has been heard previously. This, in spite of the fact that delivery is by a different person and who by all theory of chance, roulette, queueing, or whatever, will most probably introduce some new slant to an old-worn subject.

Some hold a strong belief that repetition is a good time for questioning previous information and conclusions and it is surprising how attentive listening and review will contribute to consolidation, modification or rejection of views previously held.

Having suggested this technique is most appropriate to this paper, lets have a look at some others.

Language. One of the most important facets of computer success, particularly if its a standard language. Yes - excellent for computers but humans endowed with a high degree of flexibility and even ingrained prejudice on occasions, can very successfully employ this facility in perpetuating the tower of babel.

The author is no exception. This paper could have well contained an extensive use of cliches, phrases and terms which could be recognized by some as a "mother tongue" and the rest of us would give voice to a mixed chorus of protest representing other "mother tongues".

Vendors are roundly critiscised at times for continually creating new phrases and terms which must fairly be recognized as necessary to describe the new technology that floods the marketplace without a break. In spite of the skills employed in this inventive process the more astute user occasionally recognizes a new name for something old and as a consequence users tend to take every opportunity of hearing all versions, if time allows!

Good luck with the interpretations.

Conversely vendors should be ever conscious of all the possibilities which exist in practising "mother tongue brogue" particularly as occasionally the experience and effort could be flavoured by an order.

Even through the levity hopefully a problem can be recognized. Attention to the art of good communication can alleviate it but most probably never remove it, being the people we are, complete elimination would be unacceptable and take some joy out of life.

One final comment on this point which seems to have relevance. A consultant computer designer of an advanced hardware architecure was reviewing the status of the product nearing maturity. (The point where it works) He apparently had been somewhat frustrated to this point by the lack of enthusiasm and appreciation for the elegance of his creation. His final remark - "I feel at last there is some recognition and understanding of this system, but sense and am pleased there still exists sufficient confusion to leave room for innovation."

No doubt about it, in our field innovation is necessary when communications fail.

To some extent the taking of this flexible language license introduces a potential risk which, unless pro- tected by good communication, sometimes can give rise to the misunderstood charge of misrepresentation. Is the choice of words sufficient to make the point?

In retrospect, a fair evaluation of the facts will generally identify some aspect or event where communications were not as good as they should have been. The user on one hand may feel some information of the organization should not be available to the vendor. The vendor for very proper reasons may be forced to withhold information which later the user considers to be of critical significance. However, they occur, experience has shown frankness embedded in the communication process is a useful ingredient, in resolving any impasse and importantly builds trust.

In recent times industry publications appear to contain an increase in the number of articles on the subject of contracts. The user/vendor relationship is one traditionally embodied in an agreement by way of contract and does at times become a useful document to either party. The question could be raised as to whether the postures adopted on occasions in entering these agreements run counter to the good intent of the parties in building the relationship necessary for success. Fortunately actions arising from these contracts seldom arise in the New Zealand scene and the conclusion could be that good relationships do exist or perhaps the presence of the contract provision does assist in stimulating our awareness of the consequences. The validity of this observation can only be established in places other than the conference floor.

In spite of the rigidity of legal contracts, which form the backdrop of our relationship, we appear as a group to get things done - in a majority of cases - pretty well. Often this will be attributed to ensuring that both groups of people involved in the task are operating in a manner that individual expertise is applied correctly.

Each user situation calls for a varying degree in contribution of resource and services from vendors. In the early life of an installation this is generally well defined, and has a tendancy to be a period when more reliance is placed on the vendor expertise. At later stages as user competence grows and particularly as the vendor product is moulded to user application, the vendor involvement tends to reduce and in some cases ceases completely.

This situation can arise due to the degree of competence displayed by the user due to regular intimate involvement of the product in real work-day situations, ;and which is often heightened by a feeling that the vendor personnel lack of real world experience.

On the other hand while vendor personnel do not have the opportunity of working closely in the day to day user operation, in most cases many of the products employed by users are also the basis of vendor systems. (Some users may greet this with disbelief but it is true.) In addition the vendor personnel does have the opportunity of obtaining other ideas from other user installations. Further, the vendor personnel live in an environment often described as one where "the truth changes daily" - meaning the experience of being informed of the up to date status of rapid product evolvement and techniques and not to be construed with the misunderstanding connotation.

Communications are the only way the individual contributions can be meaningfully employed in the joint interest. It would be fair comment to suggest there are occasions when this aspect of our relationship is not put to best effect - some would say "only when a crisis occurs". Good communications can and should avoid this.

Discussion of this facet of the relationship cannot help but direct a thought about the process of personnel movement between the user and vendor fields. Statistics could be expected to show a heavy bias in favour of a vendor to user migration. It would be interesting to perform a survey of Society members that could reveal a majority or certainly a substantial section possess records in their credentials and most probably some "souveniers" related to their time with the "big vendor".

Such a change on the surface appears to be natural and simple but there are most probably many instances where apparent different postures and attitudes come to be adopted in the new role.

This not to suggest anything unusual but more to make the point that such moves involve a new perspective to the user/vendor relationship. The vendor experience - hopefully good - can or should be, beneficial to the partnership. Communications ironically sometimes suffer due to tendancy of the vendor's part to assume that a "new disciple" in an installation reduces the need to maintain healthy contacts. How mistaken we can be at times. Vendors beware.

All users should be on guard to avoid "installation isolation". No, not a disease, but a situation which can occurr due to a variety of factors ranging from "far too busy, under pressure" to "no-one can teach us anything" attitude.

Vendor presentations do have their place and while great sources of information in the right "brogue" naturally have an unashamed connotation and purpose to assist in the prime objectives of the respective company. In spite of this all would agree they are good communciation mediums especially when modest lubrication by way of light refreshments is applied. (By way of diversion the initial thought for this paper originated at a Christmas function - communciations were excellent).

User groups can perform a very important function but it could be questioned whether user participation and interest is as enthusiastic as it could be. To avoid the isolation bug, to what extent is there extensive sharing of good constructive ideas, procedures and systems. Is the attitude towards the vendor constructive in terms of improving relationships for mutual benefit? Importantly, are they user promoted and run with the correct participation of the vendor representatives? At least communications should be good here as there should be "standard language". In spite of this secular interest, user groups comprising as they do of Computer Society members should assist in advancing the objectives of our organization.

Finally in the context of the Computer Society itself. Some of the attendees at this Conference will recall the apprehension expressed when the Society was founded in New Zealand, that it could become an organization dominated by the vendors.

A variety of strategies were employed by all parties to avoid the situation arising. The fact that didn't was to some extent due to communication lines which in those days had a distinct advantage of being shorter, compact in area.

Indeed it is noticeable that in spite of a larger number of vendors represented today, the user group is still very much in the majority.

For those of us who have remained in the "vendor field" exclusively over the years, the Society has been an excellent environment to develop relationships through personal contact and most probably in some way contribute to the healthy competition which we all enjoy and so continue in providing service of the users.

This is and can be profitable to all through good two-way communications.

FUTURE DATA BASE AND APPLICATION DEVELOPMENT TECHNIQUES

AUTHOR

CLIVE B. FINKELSTEIN
Managing Director,
Infocom Australia,
Highland Centre,
7-9 Merriwa St,
Gordon NSW 2072

ABSTRACT

While we have seen incredible advances in computer technology, particularly during the 1970's - with many more developments still in the laboratory indicating further advances into the 1980's - we are inhibited in our effective use of these technologies by our current application development techniques.

Data Base Management Systems (DBMS) have offered us some opportunity to manage our data more effectively, but we are still applying 1960's file and data base design techniques which inhibit effective use of data base.

The use of Data Analysis techniques to design data bases which are better able to cater for new and changing applications with minimum program modification are discussed. Data bases designed using data analysis are more likely to survive future changes in data base software and architecture.

Future data base trends towards distributed data base and back-end DBMS are discussed. The possible emergence of Relational Data base is considered, and the ability of data analysis techniques to design data bases based on today's DBMS products, but able to move later to relational data base, is highlighted.

Future application development and maintenance techniques are addressed, using today's environment of query languages data dictionaries and fill-in-the-blanks application systems as a guide. A move to an environment based on an integrated System Dictionary and Automated Application Development System, with greater potential for Security and Privacy control, is presented.

KEY WORDS AND PHRASES:

Data Base Management Systems, DBMS, Data Analysis, Relational Theory, Normalisation, Relational Model, Structural Model, Security, Privacy, Distributed Data Base, Back-End Data Base, Relational Data Base, Application Development Systems, Data Dictionaries, Query Languages, System Dictionaries, Automated Application Development Systems, Application Dictionary.

C.R. CATEGORIES:

4.0, 4.33, 4.34, 4.39, 4.9

FUTURE DATA BASE AND APPLICATION DEVELOPMENT TECHNIQUES

1. INTRODUCTION

The first twenty years of the commercial Data Processing Industry has seen unparalleled growth in the speed and capability of computers. With this growth and capability, computers are increasingly being used as a indispensable tool for business.

However, the computer industry is now at the cross-roads, where the cost of using computers is outstripping the cost of the equipment itself.

Technological advances in recent years have increased the capability, and reduced the cost of computers and terminal equipment. However, the same cannot be said of application development and control techniques. Increasing application complexity, together with the high cost of experienced DP personnel, have resulted in application development costs, operating costs, and maintenance costs exceeding the cost of computer and terminal equipment by a factor of more than two in many installations.

We have seen the emergence of data base techniques, which increase application development productivity, and enable organizations to at last gain effective control of their data resources.

However, many organizations implementing data base applications have used the Data Base Management System (DBMS) as a sophisticated data management access method rather than utilize the full potential offered by data base.

Many organizations have not designed their applications for data base - instead they have abandoned their applications to data base.

This paper briefly examines the current data base environment, highlighting areas where we can improve our use of data base techniques. Future data base developments are discussed, together with an emerging emphasis on new application development techniques which may eventually result in the demise of the programmer.

2. CURRENT DATA BASE ENVIRONMENT

From the first Data Base Management Systems (DBMS) of the late 1960's, we have seen data base become an increasingly accepted technique for proper control and management of an organization's data resources.

However, the promise of increased application development and maintenance productivity has not been realized in the great majority of installations.

The reason for this, in part, can be attributed to the complexity of some of our current DBMS products. However, a greater contributing factor has been the way in which we have applied data base to applications.

Many data base applications have been developed using, as a basis for data base design, the record formats used in previous traditional data management files. However, this presents a limitation to effective use of data base. Such data files generally may have been developed to satisfy the specific requirements of a particular program or group of programs, rather than reflect the application information requirements for the organization.

In the past, programs were developed to satisfy specific application processing requirements, and in the process of such program development the programmer also defined the necessary record formats for data files required by that program. The result of this approach is that the data content of records generally reflected the unique processing requirements of the programs, rather than the overall information requirements of an organization.

An awareness of the need for proper data analysis preparatory to data base design and implementation is emerging.

Data Analysis enables an organization to express its information requirements in terms of "data structures". Data Analysis enables stable data structures to be designed, which are most likely to survive future application changes and changes in data base software technology, while minimizing the amount of program modification necessary.

2.1 Data Analysis Overview

Data Analysis results in the development of three types of data models. These are:

- Relational Model
- Structural Model
- Physical Model

The Relational Model considers the information requirements of an organization first, and applies relational theory to the definition of data relationships. These relationships are considered as a series of tables of information, related together by common "keys".

2.1.1. Relational Model

Relational theory is well presented in the literature (Martin : 1976, 1977).

A Relational Model is developed from the business rules of an organization. These indicate the broad policy and procedures followed by that organization for the application. Data elements required by the application are identified from existing input documents and required output.

These data elements are combined in related groups - called "relations". The technique of normalization (1,2) is then applied to these relations.

Consider an EMPLOYEE relation as shown below:

EMPLOYEE NUMBER	EMPLOYEE NAME	ADDRESS	DEPENDENTS	SALARY	SKILL	CHARGE RATE	DEPARTMENT NUMBER	DEPARTMENT NAME	DEPARTMENT LOCATION	PROJECT NUMBER	TASK NUMBER

This table can be expressed in a short-hand way, with the name of the table becoming the name of a relation, and the columns becoming columns (or "domains") of the relation. The key field is identified by underlining, as illustrated below.

EMPLOYEE (Employee #, Employee Name, Address, Dependants,
Salary, Skill, Charge Rate, Department #,
Department Name, Department Location,
Project #, Task#)

Normalization is a technique which can be applied to relations as shown above, to produce fully normalized and stable relations.

Normalization proceeds through several stages. First Normal Form removes any repeating groups (Dependants above) into a separate relation.

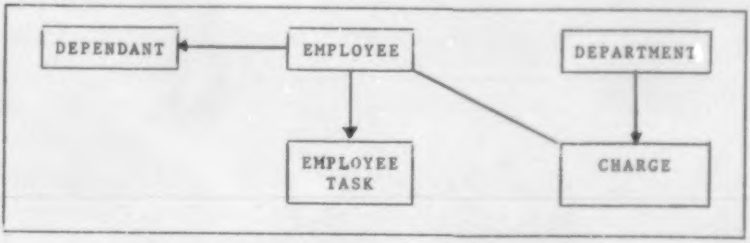
Second Normal Form removes those fields from each relation that are only partially dependent on the key of the relation.

Third Normal Form removes those data elements in a relation which are dependant not on the key of that relation, but on other data elements in the relation.

Normalizing the EMPLOYEE relation results in the following separate relations, which are in Third Normal Form:

EMPLOYEE (Employee #, Employee Name, Address, Salary, Skill, Department#)
DEPARTMENT (Deparment #, Department Name, Department Location)
DEPENDANT (Employee #, Dependant)
EMPLOYEE-TASK (Employee #, Project #, Task #)
CHARGE (Department #, Employee #, Charge Rate)

The process of Normalization ensures that data elements exist in separate relations because of their consistent meaning and use. A Relation Model can be represented diagrammatically in a Relational Map. An example of a relational map for the EMPLOYEE relational model follows.



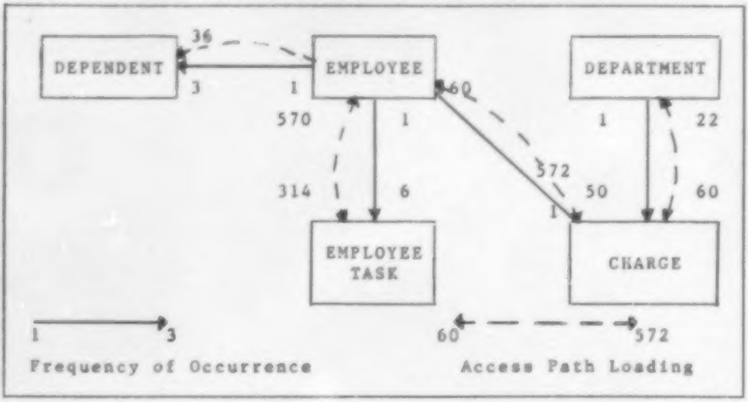
Such a relational map can be used as a basis for definition of a Structural Model.

2.1.2 Structural Model

Given the various relations defined in the Relational Model (and represented diagramatically in the relational map), the frequency of occurrence of each relation is determined for the application. The relations are then examined in terms of the various reports or queries required by the application. Such reports or queries identify the various access paths through the different relations.

The most frequently accessed paths through the relations can then be determined based on the daily volume of requests for the particular query or report.

Such an analysis produces a Structural Model, which presents the logical structuring of data for the application to ensure greatest stability and minimum future program modification. Furthermore, the Structural Model enables a relative assessment of performance of production of various queries and reports by identifying the most frequently accessed paths. An example of a Structural Model, with access path loadings follows:


DEPENDENT
EMPLOYEE
DEPARTMENT
EMPLOYEE TASK
CHARGE
36
3
1
60
570
1
1
22
572
314
6
50
60
1
1
3
Frequency of Occurrence
60
572
Access Path Loading

The development of a Structural Model, following the above approach, at no point has required a knowledge of the particular physical implementation for those data structures. Instead, data has been grouped into relations based on the input and output documents and business rules of the applications.

2.1.3 Physical Model

The Physical Model now considers the particular implementation characteristics of the DBMS to be used, in designing the data base to take best advantage of that product for optimum performance of the data base designed in the Structural Model.

. . . .

The technique of Data Analysis, outlined briefly above, is a methodology which can be used by systems analysts, data administrators and data base administrators to design data bases in the future which better reflect the information requirements of an organization. Such data bases exhibit greater stability from the effects of future application and data base software technology changes, and enable these changes to be accomodated with minimum program modification.

The Data Analysis methodology is documented in an Infocom Australia manual (1978). It is a methodology which has also been taught to User Department personnel without prior computer experience, and enables them to express their extensive application knowledge in a standardised form in terms of stable data structures. Such data structures can then be used by a data base administrator for design of a physical data base using any of the present Data Base Management Systems available today.

2.2 Security and Privacy

An area of increasing concern in the data processing industry and the community at large is that of Security and Privacy of information.

Security is defined as the unauthorized access to information. Privacy, on the other hand, refers to the "abuse" of information by people authorised to access that information.

Current Data Base Management Systems have provided a varying degree of Security control of information. The philosophy of data base is such that there is a separation of control between the application programmer and the data base administrator.

The data base administrator defines the physical data base structure, and also indicates the security controls to be applied by the particular DBMS product over that data base. The data base structure is defined in a schema using a Data Definition Language (DDL). The DDL enables passwords (or "privacy locks") to be applied to the entire data base, specific records, fields or relationships between records (sets).

2.2.1 Security

The data base administrator additionally may define a subset of the data base, containing only that part of the data base which a specific application program or group of application programs is authorized to access. This subset data base definition is referred to as a subschema. Additional security passwords or privacy locks can be applied to the various records, fields and set relationships making up the particular defined subset of the data base for those programs.

The application programmer, therefore, is aware of only of that subset of the data base defined in the subschema. He is not aware of, or permitted to access, parts of the data base outside that subset. Furthermore, the subschema may restrict the various operations which the application program can carry out against that subset data base. For example, certain records may be defined as read - only, with no authority being granted to update add or delete information in the data base.

Security is concerned not only with the prevention of unauthorized access to parts of the data base by the application programmer, but also considers the prevention of unauthorized access to the data base from various terminals in an online network, or by various terminal operators.

Such additional security can be controlled by the particular Data Communications monitor used in conjunction with the DBMS.

Transactions entered from online terminals identify the particular processing to be carried out by the computer, through the use of a transaction code for example. Such transaction codes are related by the computer to specific application programs which carry out the required processing. As discussed above the data base administrator can ensure a reasonable amount of security control to prevent application programs accessing unauthorized information.

The Data Communications monitor can control the unauthorized use of specific terminals or transaction codes (and hence access to programs and thus data for which they are not authorized) through the use of passwords. Passwords may be required to be entered in conjunction with certain transaction codes. An incorrect password will prohibit access to the program identified by the particular transaction code.

An additional security approach can be based on the entry of specific transaction codes only from particular identified physical terminals.

The above comments relate to procedures implemented to a greater or leseer degree for Security Control by existing DBMS products and Data Communication Monitors.

However, some DBMS products available today do not afford the same degree of security control (perhaps through the extensive use of passwords as described above) at the schema level - and in some cases do not even provide a subschema capability. Thus, the application programmer, knowing the structure of the data base, is free to access any data within that data base.

2.2.2 Privacy

The question of privacy relates to the abuse of information by persons otherwise authorized to receive that information. Privacy is not properly addressed by today's data base technology. The assumption is always made by existing data base approaches that the person authorized to receive the information (through security control procedures) is also a responsible person who will use that data in the way it was intended.

Circumstance of recent years has indicated that this assumption is not necessarily always correct.

Proper Privacy control is a problem with many social implications as well as computer technology implications. Privacy is perhaps one of the greatest areas of concern today, and is not adequately resolved by current technology.

The implications of future data base and application development techniques in offering the possibility of better Privacy control are addressed later in this paper.

3. FUTURE DATA BASE TRENDS

The last few years have seen a accelerated trend towards Distributed Processing, with mini computers, micro computers and intelligent terminals distributed remotely through an organization and communicating with a central host computer.

The ability of such distributed systems in processing data in a stand-alone fashion independent of the host computer has enabled significant cost, availability and performance benefits to be realized.

A recognition of this capability in the data base environment has resulted in the migration of Data Base Management Systems to distributed mini computers for processing applications in a Distributed Data Base Environment.

This section of the paper considers data base trends over the next few years, and addresses projected developments in the areas of :

- Distributed Data Base
- Back-end DBMS
- Relational Data Base.

3.1 Distributed Data Base

The next five to ten years will see further emphasis on distributed data base, first based on the larger mini computers, and then migrating down to the smaller mini computers and to micro computers.

The first micro computer DBMS announcements have been made already (Micro-Seed DBMS on Transaction Design Labs Xitan Z80 micro computer in New Jersey). Current distributed data base systems generally exhibit serious exposures in the areas of security, data integrity, deadlock detection and prevention and privacy.

As the technology develops, we will see a wider distribution of processing function than is presently realized.

Distributed data base (using full DBMS in the distributed system) will increase the autonomy of each distributed location - in its ability to manage and manipulate its own data. There will still, however, be a requirement for central control of the various distributed data base nodes, for consolidation of the various remote data base processing activity into a high-level, centralized, organizational data base.

Such an organizational data base generally will contain summary data, rather than the detailed operating data required for the day-to-day functioning of the various distributed locations.

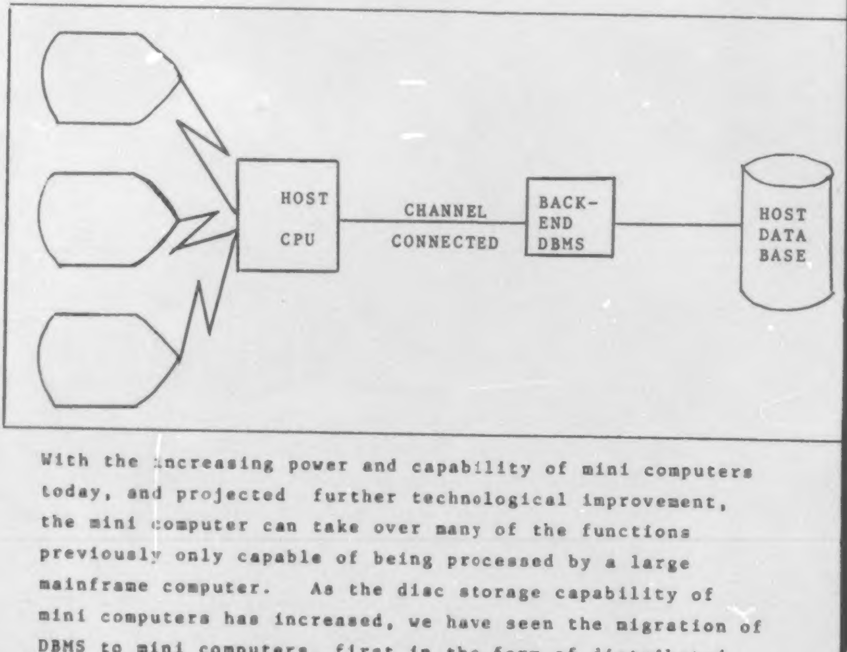
This evolution of distributed data base will tend to follow the typical management reporting levels in an organization. The distributed data base systems will provide day-to-day operating data required by first line management. This data will be consolidated and summarized for the requirements of higher-level management, until eventually the key management information required for corporate design-making is available. In the first instance, distributed data base may tend to off-load a considerable amount of data base processing to the distributed locations. This will increase the capability of the central host system to effectively consolidate the various organizational data.

While the trend towards distributed data base will increase, there will also be a parallel development towards back-end Data Base Management Systems.

3.2 Back-End DBMS

The concept of a front-end communications controller has become an accepted fact of data processing today. Such a frontend controller offloads much of the processing previously carried out by the host system, to control the transmission of data to and from remote devices.

A similar development will be seen over the next few years towards the back-end DMBS controller.



With the increasing power and capability of mini computers today, and projected further technological improvement, the mini computer can take over many of the functions previously only capable of being processed by a large mainframe computer. As the disc storage capability of mini computers has increased, we have seen the migration of DBMS to mini computers, first in the form of distributed data base systems.

Current development is underway by Cullinane and by Cincom in the migration of IDMS and TOTAL respectively from a host mainframe to a mini computer, channel-connected to the mainframe. Such a channel-connected mini computer provides a back-end DBMS capability.

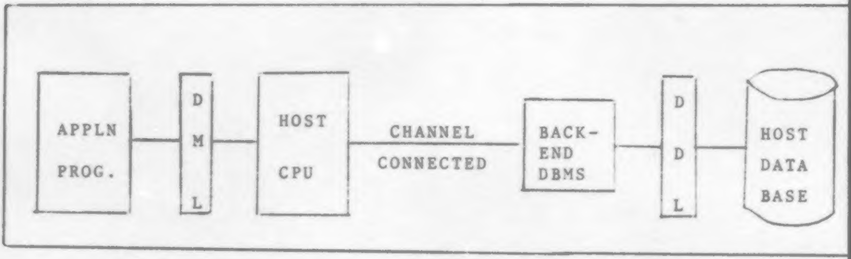
Back-end DBMS offers the advantage of uniquely designed mini computers, capable of accessing data bases in the billion byte size. Furthermore, such specialized mini computers can be microcoded to support a unique instruction set which provides processing capability equal to, and in some cases exceeding, the processing capability of large mainframes using a generalized instruction set.

The back-end system comprises the data base management system, and supports the data base according to the physical data base specifications indicated by the Data Definition Language (DDL).

The host mainframe on the other hand, processes online and batch application programs, which continue to request data base operations through the Data Manipulation Language (DML). Such DML requests are intercepted by a DML interface routine in the mainframe, and then communicated across the channel to the back-end DBMS.

The back-end DBMS interprets the DML command, accesses the data base using the DDL schema, and passes the requested data across the channel to the mainframe. Subsequent processing in the mainframe may return that data to the mini computer for subsequent update of the previously retrieved data base records.

The logical interface between the application program and the DBMS is through the Data Manipulation Language. The DML in general permits a separation between the logical accessing of a data base by application programs, and its physical organization as defined through the Data Definition Language, and represented in the schema.



This separation between logical and physical data base aspects provides a ready interface between the mainframe and a back-end data base management system.

The back-end data base management system must provide a number of capabilities to enable it to execute its function effectively. These include :

Multithreading : A multithreading or multitasking capability of a back-end DBMS enables the mini computer to concurrently process several DML commands passed across the channel from the mainframe.

Data Security : The security mechanisms of the DBMS, such as passwords, privacy locks and the logical subset capability offered by subschemas are also required in the back-end DBMS.

Data Integrity : Co-ordinated data base recovery and data base backout support must be provided between the back-end DBMS and the application programs executing in the mainframe.

Availability : An approach to increased back-end DBMS availability is to consider the use of multiple back-end processors. Such multiple processors function independently of each other, and each can fuctionally access any part of the data base, and so take over the fuction of other failed processors. With the increased capability of micro computers, we can see the emergence of multiprocessors implemented using microcomputers for many DBMS functions that are currently provided in one system.

3.3 Relational Data Base

One of the key advantages of data base technology is that of data independence. Data independence in current DBMS products has been implemented to varying degrees, enabling application programs to consider the logical processing against the data base, without concern for the physical organization or access to the data base.

While data independence is generally been used as a collective term, we should consider it broken into components - that of :

- Logical data independence
- Physical data independence

Physical data independence is well supported by most of the DBMS products available today. Through the Data Definition Language (DDL) the physical organization, physical structure and access methods used to implement the data base can be defined by the data base administrator.

The capability of many of today's DBMS products to offer logical data independence, on the other hand, is more limited. Logical data independence is only achieved when the application program can view the data base in a way best suited to the application processing requirements, without any concern for the physical structure of the data base.

Recognizing the limited capability of current DBMS products to achieve this logical data independecne, Codd in 1969 proposed the use of Relational Data Bases to overcome these limitations.

A Relational data base considers the data exists in the form of two-dimensional tables. Such tables contain multiple columns (which can be thought of as similar to the various fields in a record.) The rows of the table represent the many "data base records" making up the entire data base. Access to an appropriate row in the table is made on the basis of a unique key, which identifies the appropriate row in the table of the record exhibiting that key value.

The use of tables is an accepted and easy to understand method of data access by users. However, the data processing industry has become quagmired in its own complexity.

Tabular forms of data representation are natural to users, and disarmingly simple to the Data Processing Community. Hence, the interest in Relational data base. Considerable research has been carried out in relational data base technology, variously reported in the literature (Martin: 1976, 1977; Infocom: 1978; Zloof: 1976).

Relational theory permits the concise specification of a relational algebra and relational calculus to indicate the accessing and manipulation of relational data bases. Martin (1976, 1977) provides a useful summary of Relational data base.

Associated with relational theory is the concept of normalization. Normalization, as discussed earlier in this paper, results in the resolution and separation of the various data used in an organization into stable data structures, which reflect the business rules of that organization.

The technique of Data Analysis, also covered earlier in this paper, is enjoying increasing acceptance as a prelude to data base design. Data Analysis is fundamentally based on the application of relational theory to information analysis and design.

While many prototype Relational Data Base systems are under development in various Research Laboratories throughout the world, at the time of writing this paper there is currently no commercially - available Relational Data Base System.

Current computer technology is an inhibiting factor to the development of Relational Data Base Systems which exhibit adequate performance. The availability of associative memories and parallel processing techniques, together with massive main storage availability in computers will enable much of the promise of relational data base to be achieved in the future with satisfactory performance.

The development of new generalized end-user query languages using relational data base has been reported in the literature (Zloof : 1976).

Relational data base and generalized query languages offer potential in the future for easier access to information without the complexities represented by current DBMS products.

4. FUTURE APPLICATION DEVELOPMENT TECHNIQUES

The preceding sections of this paper discussed future data base trends. This section examines parallel trends in application development and maintenance technology.

The productivity of application development and maintenance is the single and most inhibiting factor in more effective utilization of computers. We are seeing today the emergence of a number of techniques which offer promise of improving this productivity, through greater automation of a number of development and maintenance tasks using computers.

Current techniques which offer a pointer to future developments are :

- Online Application Development Systems
- Data Dictionaries
- Query Languages.

4.1 Online Application Development Systems

There is presently available a number of application packages which are oriented towards the easy development of online applications. Generally these systems are structured around the definition of online terminal screen formats, with associated editing and manipulation of various data fields on terminal displays or in data file records.

An example of such an online application development system is Display Management System (DMS/VS), a program product supported by IBM.

Such online application development systems generally involve the specification of terminal display formats using fill-in-the-blank coding techniques. Provision is made to relate specific data fields on the screen with the appropriate data fields in disk files, either for enquiry applications or online update applications. A capability is also available to indicate the transfer of application logic from screen format to screen format throughout a defined application system.

As well as permitting access to standard data files, some of the systems also enable access to Data Base Management Systems. The interface with the DBMS is generally defined at a high level, with reference to appropriate records and fields made by name, not requiring a knowledge necessarily of the physical data base structure.

4.2 Data Dictionaries

Data Dictionary/Directory Systems (DD/DS) have been developed to assist the Data Processing department to manage the data resources of an organization more effectively. These data resources are the "raw materials" of the DP department. Not only must the DP department keep details of the various data fields and records used by the organization, but the various application programs and application systems which utilize that data must be noted as well. Thus there is a requirement for cross-referencing between data resources and application programs and systems. From this cross-referencing can be derived "where - used" information, which enables the DP department to assess the impact of a change in specific data fields, for example, on all of the various application programs which access those fields.

Initially, Data Dictionary/Directory Systems were developed manually, to standardize on the descriptive information maintained for each data field and record in an organization. As the complexity of applications increased, and with the emergence of data base management systems, this Data Dictionary capability has been automated.

There are a number of Data Dictionary/Directory Systems currently available as separate program products, developed and marketed by DBMS suppliers and independent software houses alike. Some of these products are themselves based on the use of a DBMS for data management and organization. Typical packages in this category include : DB/DC Dictionary (IBM), TOTAL Data Dictionary (Cincom Systems), Integrated Data Dictionary (Cullinane) and Control 2000 (MRI Systems).

Data Dictionary/Directory Systems which are not dependent on a DBMS include Lexicon (Arthur Anderson), Data Catalogue (Synergetics), DATAMANAGER (MSP) and PRIDE/LOGIK (M. Brice Associates).

We have seen Data Dictionary packages evolve from "passive" packages - which require specific input of all information to be maintained by the Data Dictionary - to integrated packages which accept existing data definitions as documented in application program structures (COBOL or PL/1) or in DBMS Schema definitions. Use of program structures or DBMS schemas avoids the separate input definition step required of passive Data Dictionaries.

Similarly, following a Data Dictionary change, integrated packages produce as output the changed program structures (COBOL or PL/1) or DBMS schemas.

Some Data Dictionary packages also provide an online interface, supporting enquiry and update of data dictionary information.

4.3 Query Languages

The need to better address the information requirements of management in resolving the request for information often of an ad-hoc nature, has seen the emergence of a number of generalized Query Languages.

Query Languages are now available for use with all of the current DBMS products, and can be used to extract that information satisfying particular selection criteria, sort and manipulate that information and report it in specified report formats. The currently available query languages use a variety of techniques, ranging from fill-in-the-blanks coding techniques to English-like statements requesting the various operations to be performed by the Query Language Processor against the data base.

Additionally, some current Query Language Processors permit information to be extracted from standard data management files as well as data bases.

The techniques discussed above - of Online Application Development Systems, Data Dictionaries and Query Languages - are indicative of future trends towards more automated application development and maintenance techniques. The emphasis in each case is on use of the computer to maintain an "inventory" of data resources together with information required to extract necessary information from that "inventory" to satisfy application and user demands. The use of data base, together with data dictionaries provide management and control of application data. Online application development systems permit the easy development and subsequent maintenance of online applications. Query languages permit the easy extraction of information to satisfy one-time queries or report requests.

The above capabilities represent a future trend towards the development and use of :

- Systems Dictionaries
- Automated Application Development Systems.

4.4 System Dictionaries

The current management of computer data resources is fragmented, with the same data sometimes defined in many different ways.

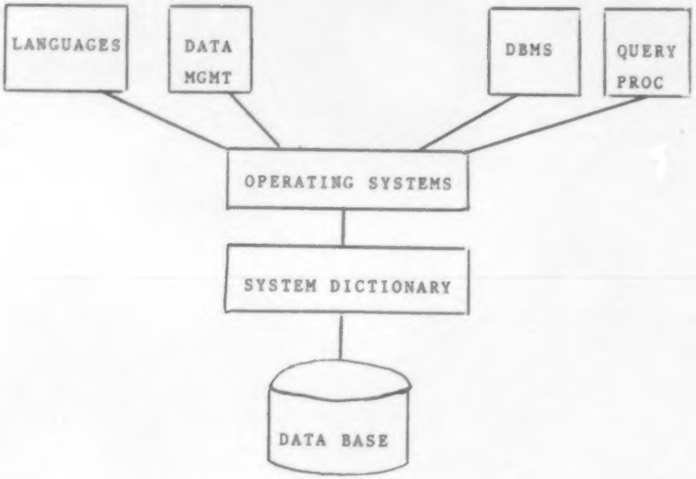
Thus data must be defined as records, formats and structures for language compilers, as records and files for operating system control, as fields and records for data base management systems, and as fields and records for query languages.

This multiple definition of the same information is in the first instance inefficient, and in the second instance a considerable source of error. The same data may be defined in many different places in an installation. Changes to that data must be reflected in all of its definitions. However, the amount of detail required in each definition varies, and the likelihood of incomplete incorporation of data changes is high.

We have now evolved to the present disjointed data resource control environment from several directions (languages, operating systems and DBMS). The need is already evident for a single, integrated System Dictionary, which is the central repository of information describing the various data resources used by an installation.

This System Dictionary should be the key, central control point of data resources, providing information required by language compilers, operating systems and DBMS. A change occuring in the definition or use of a particular data field would be incorporated in the System Dictionary, and thus immediately available to all system components which access that dictionary.

Such a System Dictionary then becomes a system-wide data base of information.



However, the path to this environment will not be easy, and may not occur until well into the 1980's because of its impact on existing investment in current operating systems, language compilers and data base management systems. We may, however, see an evolution to this central System Dictionary, through the emergence of Automated Application Development Systems.

4.5 Automated Application Development Systems

The present online application development systems described previously do bypass the need for a considerable amount of application programming in the acceptance of terminal input, the accessing of data file records and manipulation of data fields, and the preparation and display of terminal output information.

However, present online application development techniques are not fully integrated, and generally require the additional coding of application program interfaces to permit full data editing, conditional logic and processing to be carried out.

We are seeing the emergence of Automated Application Development Systems, which are structured around Application Dictionaries, and permit the online definition of screen formats, conditional logic and processing.

Such online application definition then results in the automatic generation by the system of the necessary application programs to carry out the specified screen and file management, with the required conditional logic and processing.

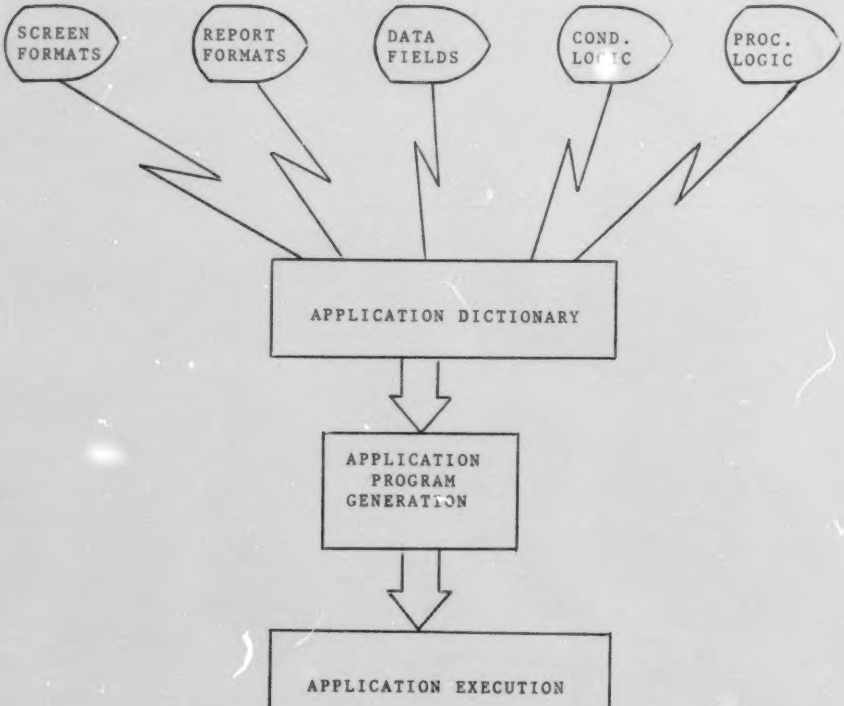
Such an approach bypasses the need, for many applications, of detailed program coding. Instead, the system analyst, or suitably trained user department personnel can develop at a terminal the required screen formats for an application. The appropriate conditional logic to be applied to specific data fields on a screen display or in a data file record is also specified, and then the necessary processing to be carried out to satisfy the application requirements is indicated.

Fundamental to such an approach is an Application Dictionary. This Application Dictionary stores all defined input and output screeen format, conditional logic and processing specifications for subsequent use in application program generation. Additionally storage of these specifications in an Application Dictionary enables application maintenance to be applied rapidly and accurately, using previous definitions stored in the dictionary - and changing only that necessary to effect the required application modification.

Following application definition, or application maintenance, the various defined screen input and output formats, report formats , conditional logic and processing statements are used by the Automated Application Development System to generate the necessary application programs, usually in a high-level language such as COBOL or PL/1.

This approach bypasses the need for detailed application coding, and utilizes the capability of the computer to translate application definitions into the necessary programs to carry out required processing. Such generated application development programs are executed by the Automated Application Development System under control of a monitor which provides much of the common data entry and editing functions for applications, with release of completed batches of data for subsequent update processing, report preparation and enquiry. In an online environment, such application monitors also provide for immediate online editing and update of relevant data fields.

AUTOMATED APPLICATION DEVELOPMENT SYSTEM


SCREEN FORMATS
REPORT FORMATS
DATA FIELDS
COND. LOGIC
PROC. LOGIC
APPLICATION DICTIONARY
APPLICATION PROGRAM GENERATION
APPLICATION EXECUTION

In such an Automated Application Development System environment, the systems analyst is not concerned with the physical consideration of application logic or data storage. Application requirements are defined in application terms rather than computer terms, and data is referenced by name. This assumes, therefore, that the Automated Application Development System is able to identify all data fields used in an installation for application definition and maintenance purposes, and subsequently access and operate using that data during actual production processing.

This capability implies a highly-integrated environment, offering a consolidation of capabilities presently provided separately by online application development systems, data dictionaries and data base management systems.

As Automated Application Development Systems evolve, we will see firstly the development of an integrated application definition and data dictionary system, with interfaces provided to standard data management files and the most common data base management systems in current use.

This evolution will permit a significant productivity improvement in application development and maintenance while still providing an interface to existing application systems through current data management files and DBMS data bases.

As the industry moves to a more productive development and maintenance environment, the need for detailed management and control of data through DBMS products will change in emphasis. We will therefore see a greater integration of the fuctions currently provided by DBMS products into an all-encompassing Application Development System. The computer will then have complete responsibility for the allocation of data fields to appropriate sections of an installation - wide data base, and the location and retrieval of those data fields for processing when required.

Such central control of data resources by computer offers considerable promise in the areas of the Security and Privacy of information.

4.6 Security and Privacy Trends

Part of the current concern regarding Security and Privacy of information is the multiplicity of definition and access to data using present computer and data base technology.

A future move to integrated System Dictionaries will place the control for definition and access to data at one point. This will permit greater security control to be exercised by the System Dictionary, as only one definition of data and one copy of data will exist.

Proper security control over multiple copies of keys to a house is more difficult to enforce than control over one master key. Similarly, System Dictionaries offer an opportunity to use only one "master key" and so more completely prohibit unauthorized access to information.

Automated Application Development Systems offer promise for greater Privacy control in the future. In conjunction with a System Dictionary (or an interim Application Dictionary) the computer may be able to determine whether the application being defined and executed by the Automated Application Development System is consistent with the normal use of that application. Inconsistent definition or access to information in an application by an authorized person may indicate a potential Privacy exposure. Such apparent "abuse" of information by authorized persons can be drawn by the computer to the attention of management, to validate - with the person - the use to which that information will be put.

The determination by the computer of "inconsistent" use of data and applications can be based on prior definition in the installation of the "normal" ways in which that data may be defined and accessed.

It will probably never be possible for the computer itself to determine whether a Privacy exposure is occuring. However, the computer can determine whether data is being accessed outside its normal defined context, and so bring that event to the attention of management who can carry out further investigation - and so determine whether in fact the particular activity identified is valid or instead a breach of Privacy.

5. CONCLUSION

As the computer industry moves to a more productive development and maintenance environment, the need for detailed management and control of data through DBMS products will change in emphasis. We will therefore see greater integration of the functions currently provided by DBMS products into an all-encompassing Application Development System. Such a system will ensure that the computer has complete responsibility for the allocation of data fields to appropriate sections of an installation - wide data base, and the location and retrieval of those data fields for processing when required.

Of course, such a generalized approach will require a great deal of processing and data storage capability. However, technological improvements in hardware and storage offer promise for the future - permitting rapid access to massive amounts of data not economical using today's technology.

Until the industry can move to such an automated application development environment, the effective use of computers will remain in the horse and buggy era of today. In the same way as the automobile industry moved away from the labour - intensive environment of the early 1900's with custom-built and maintained motor cars - to the assembly-line technology of today, so also must the Data Processing Industry move to a more productive environment.

We can no longer afford the luxury, or cost, of custom-developed application programs, with a high development lead time and cost, and consequent maintenance bottlenecks.

Hardware technology has already outpaced our ability to effectively utilize it with current application development and maintenance techniques.

We must change if the industry is to progress !

BIBLIOGRAPHY

1. James Martin (1976) : "Principles of Data Base Management" Prentice - Hall

2. James Martin (1977) : "Computer Data Base Organisation" Prentice - Hall

3. Infocom Australia (1978) : "Data Analysis - The Key to Data Base" Infocom Australia.

4. Infocom Australia (1978) : "Introduction to Data Base and Distributed Processing" Infocom Australia.

5. Moshe M. Zloof (1976) : "Query-By-Example : The Invocation and Definition of Tables and Forms". IBM Thomas J Watson Research Centre.

A RELATIONAL MODEL OF AN ACCOUNTING SYSTEM

AUTHOR: DR IVAN F. JACKSON
The University of Otago.

ABSTRACT:

This paper describes a perception of commercial data processing systems from a different viewpoint - that of a database designer interested in simple communication with the data stored in the computer system. In the first paper the author describes the role of the data dictionary and discusses the less widely known relational model of data. He shows how the two concepts can be merged to the overall satisfaction of both designer and the user.

The second part of the paper describes research conducted by the author in the above areas at the University of Otago. The concept of a relational database structure is introduced and the author explains how complex relationships can be broken down into comparatively simple forms. In particular, the approach is applied to a general ledger accounting system and the author shows how the development of the system can be substantially simplified, how reports can be easily generated, and how the relational approach can be used to resolve quite complex accounting inter-relationships.

KEY WORDS AND PHRASES

Accounting System, Relational Model, Information System, COBOL, User, Data Base, hierarchial structure, network structure, data base management systems, data dictionary, data base administrator, data analysis, chart of accounts.

C.R. CATEGORIES

3.52, 4.20, 4.33, 4.34.

A RELATIONAL MODEL OF AN ACCOUNTING SYSTEM

by

Dr Ivan F Jackson
The University of Otago

1. INTRODUCTION

A major purpose of a corporate information system is to provide an accurate and relatively efficient model of the significant resources of the enterprise. Efficiency is important as the organisation expects efficient usage of the resources it owns, as well as those which are modelled, be it cash, parts, orders or customer information. Designers of information systems certainly attempt to design efficient systems, both from the point of view of the computer itself, and from the user. A desire to increase the efficiency of the machine has lead to the development of sophisticated operating systems, with virtual memory, dynamic allocation of resources, the concept of virtual machines, the use of control storage, firmware, etc. A quest for efficiency on the part of the user has lead, through assembler languages, to the development of the higher level languages such as FORTRAN and COBOL, to the use of text editors, forms generators and on-line data entry and enquiry systems. All these have helped in the area of user efficiency, or if it is preferred, user-system communication.

It is reasonably safe to say that the improvement in a user's communication with his machine has been evolutionary rather than revolutionary. Software development is still a slow, expensive and painstaking process, often costing as much as the hardware itself, yet much less reliable. Languages such as COBOL are hopelessly tied in with data lengths, data types, position of decimal points and general operation at a character level. A designer cannot seriously propose that he can model an organisation's resources on a sequential or even indexed sequential based system. In fact he does not attempt this and relies instead on file searches and sorts to obtain data in a form suitable for the unsophisticated user.

A trend towards a better modelling of an organisation's information and improved communication with that data is through the use of data base management systems. The efficiency of the computer itself can be improved because data redundancy is reduced, better security can be maintained and multiple indexes enable data to be accessed on a number

297

of data criteria. The efficiency of the user is improved as the data is maintained separately from the programs which can now be less complex; and in some systems, users can enquire into the file not through a program but through simple query-type languages. Even with a data base management system, however, a requirement still exists for a design approach that increases the design efficiency while not sacrificing computer or user efficiency. This way communication with the computer can be improved, not only for the user, but also for the system designer and programmer. This paper explores how this might be done using a relational data base approach and describes a research project along these lines conducted at the University of Otago.

2. DATA BASE STRUCTURES

One of the more difficult problems faced by programmers and analysts when being introduced to data base technology is to grasp the concept of structuring data. In structured files data is maintained at different levels with rather complex linkages between levels. The COBOL language introduces the level-number concept where data is progressively broken down into higher (or depending on your point of view, lower) levels, each of which may be referenced directly. Regardless of the addressing mechanism, however, an '01' level record can still be written to a flat length of magnetic tape, so that data is not really structured in a physical sense. Most data base management systems structure data in a physical as well as logical sense so that the storage device can no longer be tape as it must be able to be physically addressed at different levels.

If we were to revert to the early EAM days of data processing the most common storage medium was the card deck. With a card deck data could be stored in fixed length fields, the same format being used on each successive card. Today people who work with databases like to refer to these decks as flat files. Flat files are easy to understand in concept and from a data processing viewpoint they have certain distinct advantages, not the least of which they lent themselves to relatively simple processing. Card files were not, however, particularly effective in that they did not use electronic computers efficiently, nor were they particularly easy to handle, especially for large files.

Before very long the more sophisticated card user found that 80 columns of information was insufficient for many data representations and so he moved towards multiple format records, with a 'master' card followed by a number of 'detail' cards. The master card of each series quite often had a different format from the following detail cards. In fact, what happened was that data associated with each record was 'partitioned' into the master card and the detail cards. The move was actually from a 'flat' file to a structured file, with the master record sitting as it were on the top of the deck, and the detail cards sitting at a lower level. The structuring is even more apparent if there are several different kinds of detail cards. The detail cards were often processed separately from the master cards and sometimes were only merged together ('collated') as a last phase of processing.

With the widespread introduction of magnetic tape units onto computers many card-based systems, including the master/detail records systems, were transfered directly to tape. While this may have been an expedient measure it was subject to the criticism that the converted systems were merely 'card-based' and did not make efficient use of the new recording medium. This criticism was countered by a move to unitised 'variable length' records with a 'master' component and a variable number of 'detail' components. With the continuing of the two types of records, however, the flexibility inherent in the ability to process the 'detail' data was lost. Thus, to some extent, while data processing took an overall step forward, in some respects it also took a step backwards.

With the introduction of disk storage the 'detail' records could again be separated and it became possible, in practical terms, to represent data in a hierarchy with one master record and several detail records, some of which in turn had further detail records below them. Typically a user addressed the master record and then followed 'downward' pointers to each of the detail records, and then further pointers down to the lower level detail records. Data was now partitioned both logically and physically and separate processing was possible. For instance in a savings bank the master records, grouped together into a 'master file', could contain customer information (name, address, etc.) and pointers to a 'savings files', containing details on particular savings accounts, to a mortgage file, term loans file, etc.

In some applications it was found to be more convenient to have a pointer from the master record to the first detail record, then from the first detail record to the second, from the second to the third, and so on, perhaps completing the chain with a pointer back to the master. An example might be a number of individual orders chained or linked to a given customer. It may be required, for instance, to address (i.e. 'access') the file by order number as well as by customer. It may even be required to extract information using the customer name as the 'access key'. Such a file organisation is said to be a network structure. In a network structure not only do pointers go 'down' (i.e. from master to detail), 'across' (from detail to detail) but it may be desirable to have one detail pointing to several master records, a feature not allowed in a hierarchical structure.

The increasing complexity of such structures has not made for an easy understanding of data base technology and may in fact have discouraged potential users. A group of people, including this writer, have therefore become interested in the relational model of data as a method of reducing the complexity of data representation as well as making it easier for a user to utilize the system.

In the relational approach, in essence there is a reversion to the original concept of 'flat' files. It is comparatively easy to visualise a tub file of punched cards, each card representing a series of data fields. For instance in a parts file the fields could represent a part number, part description, vendor number, quantity on hand, recorder quantity and unit cost. As orders are filled cards would be withdrawn by hand from the tub file and the quantity-on-hand field updated. Such a system could be readily converted to a disk based system with each card image capable of being accessed directly. Although the old characteristics of the tub file have been retained, a user now has the convenience of working with magnetic images rather than with cards.

In a relational data base model the parts file can be represented as a 'relation', e.g.

PARTS (PART-NO, DESCRIPTION, V-NO, QOH, R-PT, COST)

Clearly this representation tells us a considerable amount about the file. It tells us the file is called 'PARTS' and that it consists of a number of named data items. It also tells us, using an underlining convention, that the file is accessed on the key PART-NO. One does not have to be tutored in the new mathematics to appreciate that each data item represents a 'set' of data elements. It follows that the relation presented above can be broken down into a flat file which is in turn represented as a table, as shown in Figure 1.

PART-NO	DESCRIPTION	V-NO	QOH	R-PT	COST
121	NUTS-HEX	4	73	50	2.87
129	NUTS-WING	4	34	30	3.95
273	BOLTS-8mm	7	81	60	5.73
etc.					

Figure 1. Concept of a 'flat' file.

Each entry corresponds to a record with each element corresponding to the contents of an individual field.

With the relational representation and the applicability of set theory it would seem plausible that a special kind of data algebra could be used to manipulate data and that potentially a computer could be built that would have relational operators built into the hardware. It could manipulate data in a manner similar to that used by arithmetic and other 'hard-wired' operators. Early work on a relational algebra was done by Codd[1] of IBM while a number of experimental relational implementations have been reported[2].

An interesting way of presenting a relation is, in the form of a dependency diagram - see Figure 2. The various elements in the relation (called 'domains') are shown to be 'functionally dependent' on the set of part numbers. Figure 2 should put at rest the minds of those readers who are familiar with the more conventional data base representation.

It can be legitimately argued, however, that it is a rather simplistic approach to represent all data as flat files. This criticism is valid insofar as it goes. It can be shown, however, that by a process called decomposition files are split up and in some cases redundancy introduced, and all files can be broken down into flat files.

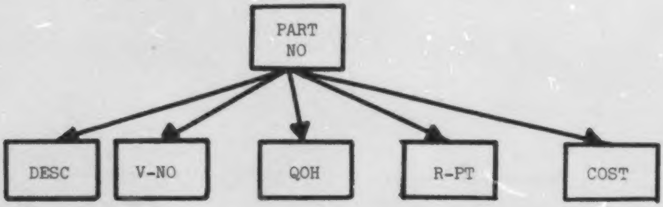


Figure 2. Functional dependency diagram.

In the situation where redundancy is introduced consider a parts file organised on the basis of vendor. The original file could be represented as shown on the left hand side of Figure 3, and with redundancy introduced as shown on the right hand side of the same figure.

Vendor No	Part No	Qty
V1	P1	3
	P2	1
	P3	4
2	P3	3
	P4	4

BEFORE

Vendor No	Part No	Qty
V1	P1	3
V1	P2	2
V1	P3	4
V2	P3	3
V2	P4	4

AFTER

Figure 3. Introduction of redundancy.

One of the problems of the original form is that it can lead to a loss of data integrity. For instance if the record containing P1 was deleted, it would not be possible to determine the supplier of P2.

The process of decomposition is quite complex and will not be discussed here. For illustrative purposes, however, consider two relations:

Product (Product No., Product description, Quantity
on Hand)

and Orders Received (Product No., Order No., Export status,
Customer number, Customer priority, Order quantity).

for convenience these may be designated
P(PN, PD, QOH)

and OR(PN, ON, X, CN, CP, Q)

It can be shown that these can be reduced to
(PN, PD, QOH)

(ON, PN, CN)
(PN, Q)

(CN, CP, X)

which actually consists of three separate 'data structures' in which deletion and other processes (i.e. the addition of a new customer) does not affect the integrity of the data base. In the decomposition process the original relations have been reduced to a group of four 'flat' files, each with consistant properties.

3. DATA DICTIONARY

The growth in complexity of data structures has lead to the introduction of data base management systems ('DBMS's) to the system developer in his interaction with the computer. Examples of DBMS's are IBM's IMS/VS and Burrough's DMS-II. Such systems are in essence extensions to existing procedural languages (e.g. COBOL, PL/I). Other systems are 'stand alone', e.g. Hewlett-Packard's IMAGE or Infomatic's MARK-IV, and generally have an executive oriented 'query' language for extraction of information using English-like sentences.

Most DBMS's use the concept of a data dictionary which is similar in concept to an ordinary dictionary but contains information about the definition, structure and usage of data, but does not contain the data itself. Each entry in a data dictionary contains the name of each data element, its definition (e.g. numeric or alphanumeric, length, etc.) and its relationship to other data. This last property clearly has particular significance in the relational model.

In conventional data processing systems it is not uncommon for several programmers to be using the same files. As long as the data is addressed directly from application programs there is no problem in each programmer calling the same field by a different name. Under such circumstances a voluntary standardisation as to naming conventions may be imposed in the interests of consistant documentation, programmer interchangability, etc. In a DBMS, however, standardisation of names becomes essential as the management of the data and of the data base itself is physically separated from the programs. The management of the data becomes a separate data processing function, commonly performed by a 'data base administrator' (DBA).

In recent years the importance of the data dictionary has been widely recognised and data dictionary systems are now available (at a cost between $10,000 and $25,000) for use independent of the DBMS systems. Typically such systems have a data definition language for defining entries in the data dictionary, a data manipulation language for inserting, deleting and modifying entries, processing routines for validating input, and an RPG-like language for preparing output. Some systems also provide for a cross reference listing showing, for each entry, all uses of the particular data item, in terms of where it fits in the data base, and the programs using the data item. A reference discussing the role of the data dictionary has been published on Datamation[3].

The implications of the use of data dictionaries have a subtle effect on the role of the systems analyst. In fact it opens up a whole new field. This field has become known as data analysis. Using analysis, a system is designed from the 'bottom up' instead of from the 'top down' as in conventional systems analysis. Systems analysis tends to work from organisational objectives, divided into functional areas. These are subsequently ranked in terms of value, cost and priority. The details of each processing function are defined and finally the data requirements of each function are determined. In data analysis the data in an organisation is considered as a resource of that organisation. Steps are taken to define and quantify the various data elements and where necessary standard names are assigned. The data is then mapped onto storage in the form of a data base in a manner likely to optimise the usage of the data. The remaining systems analysis functions are then developed.

4. AN ACCOUNTING MODEL

In investigating the applicability of the relational data model, especially when applied to a small computer, the writer was stuck with the similarity between the relational model expressed as a dependency diagram, and company's chart of accounts. At the same time the writer was investigating matrix accounting methods as a technique for implementing a general ledger system on a mini-computer so it seemed logical that the two studies go hand in hand.

In addition to the above factors it occurred to the writer that when set theory was applied to the relational model of data it was generally assumed that the data files consisted of a large, or at least considerable, number of records. This is the general case. There seemed however, no reason why a set could consist of just one entry in a record. A given record, like the file, would be considered to consist of a number of sets of data, each set with only one entry. The access technique for a unary set is easy with a computer, the set is simply a variable. However, if the set was a conventional one, representing a number of different values of a given variable, then the basic logic would remain the same and only the access method would be different. In the unary case the value of the variable could be addressed directly; in the conventional case the particular record could be obtained through file, (e.g. disk) addressing.

Reverting to the chart of accounts' model, a typical component of a chart of accounts, expressed in original chart form, is shown in Figure 4. If the chart of accounts is viewed as a dependency diagram, similar to that shown in Figure 2, then it is possible to specify a series of functional relationships along the lines of

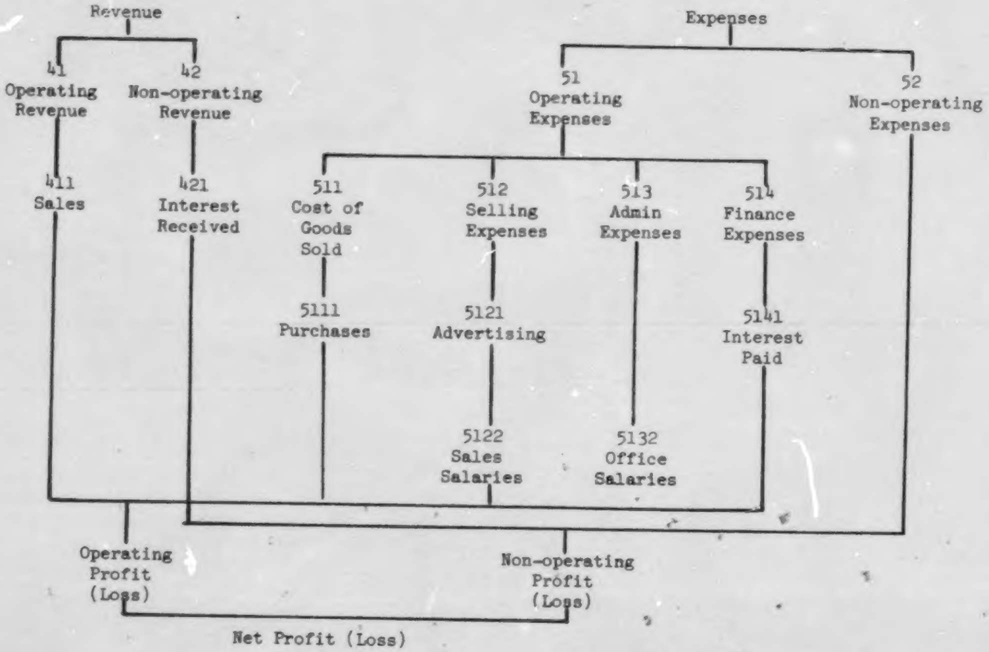

Revenue
41
Operating
Revenue
42
Non-operating
Revenue
411
Sales
421
Interest
Received
Expenses
51
Operating
Expenses
52
Non-operating
Expenses
511
Cost of
Goods
Sold
512
Selling
Expenses
513
Admin
Expenses
514
Finance
Expenses
5111
Purchases
5121
Advertising
5141
Interest
Paid
5122
Sales
Salaries
5132
Office
Salaries
Operating
Profit
(Loss)
Non-operating
Profit
(Loss)
Net Profit (Loss)

Figure 4. Classification of Revenues and Expenses

REVENUE = f(OPERATING REVENUE, NON-OPERATING REVENUE)

and

EXPENSES = f(OPERATING EXPENSES, NON-OPERATING EXPENSES).

It is then possible to define a further breakdown:

OPERATING EXPENSES = f(COST OF GOODS SOLD, SELLING EXPENSES,
ADMIN EXPENSES, FINANCE EXPENSES),

and in turn, for example,

SELLING EXPENSES = f(ADVERTISING, SALES SALARIES).

In fact what has been identified in Figure 4 are two data structures, one relating to 'Revenue' and the other related to 'Expense'. Within each structure, a number of relationships may be defined and these relationships may be seen to belong to successively higher 'levels'. The situation may perhaps be clearer in the 'NET PROFIT' structure shown in Figure 5. The various levels are shown in the figure. It will be noted that the values for lower ordered levels may be obtained before the higher ordered levels related to a particular variable are also obtained. Thus it would not be possible to compute GROSS PROFIT unless COST OF SALES, situated on a higher level, had been first computed. Likewise to compute NET PROFIT it is necessary to know BEFORE TAX PROFIT (which in turn requires GROSS PROFIT and EXPENSES) and TAXES.

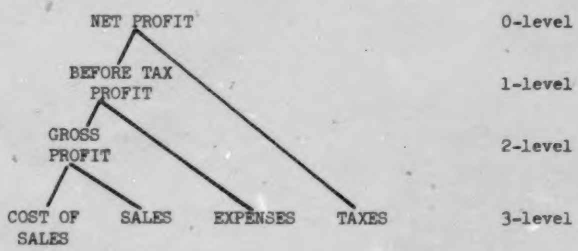


Figure 5. The 'NET PROFIT' structure.

The structure shown in Figure 5 may be expressed as relations, but it is obviously convenient to represent it as a series of specific kinds of relations, namely equations:

NET PROFIT = BEFORE TAX PROFIT - TAXES (1)
BEFORE TAX PROFIT = GROSS PROFIT - EXPENSES (2)
GROSS PROFIT = SALES - COST OF SALES (3)

It can be seen that there is a definite order of substitution in the structure, with equation (3) evaluated before (2) and (2) evaluated before (1). The variables on level 3 of the structure shown are, in fact, 'predetermined' (in practice COST OF SALES would be made up of a number of higher level factors as would the other variables) and could be stored in a disk file and accessed directly. Variables at a lower level cannot be accessed directly and so the structure must be decomposed in the same sense as that described for the relational data base approach.

In the structure shown in Figure 5, the 'root mode', namely NET PROFIT, serves to identify the structure. In the study conducted at Otago a chart of accounts, as used by a form of chartered accountants, was first structured (i.e. drawn out by hand) and then coded in the form of relationships. Root modes identified in the structuring process were:

ASSETS	(6 relationships in 30 variables)
LIABILITIES	(8 relationships in 31 variables)
CAPITAL	(4 relationships in 14 variables)

constituting the balance equation while for the revenue statement the following root mode applied:

NET PROFIT (31 relationships in 133 variables).

In addition to the above other structures were identified for form ccounts, sole proprietors, partnerships, etc.

5. A RELATIONAL APPROACH

The various relations contained in the accounting system, numbering 90 in all, were coded into what is essentially a sequential file, but which can also be addressed by record number, one relation per record. The variables were coded as 10-character mnemonics(originally 6 characters was tried but with so many variables it was sometimes difficult to obtain sufficient differentiation). Also included in the relation was the conventional account code. A typical record would appear as follows:

CASH = CASH O/H 510 + PETTY CASH 512
+ OUTSTD LDG 514 + SAVINGS 516.

A series of 3 files were derived from the basic relationship file. These are the accounts file, the associated matrix file and the cross-reference file or data dictionary.

1. Accounts file. This file is similar to that mentioned in a conventional general ledger system except that it contains no descriptive information nor account balances. Information contained in the file, organised indexed sequentially by account number, is

   Account code (record key)
   Account mnemonics
   Relative matrix location
   Root mode (relational ID) for structure
   containing variable

   Transactions may thus be coded in a conventional manner and may be entered directly into the system by account code. Only those variables from nominated structures (i.e., farm accounts, partnerships, etc., can be excluded) are included when the account file derived from the original relationahip file.

2. Cross-reference file (data dictionary). This file, as its name implies, is organised alphabetically (actually indexed sequentially) by variable mnemonic and contains one entry for every variable named in the relations file. Information contained in the file includes:

   Mnemonic
   Account number if present, alternatively
   relative position in relations file
   Descriptive entry (for financial statements)
   Relative matrix location

Transactions can thus be coded by mnemonic (i.e., RENT instead of 384) and the amount can be posted directly to the appropriate account in the matrix file. The cross-reference file is also derived directly from the original relations file.

3. Matrix file. The concept of a matrix file was developed as a result of research (performed in conjunction with this project) into the possibility of adopting matrix accounting concepts to a mini-computer. Essentially the file consists of a series of records equal in length to the total number of records in the accounts file. Thus if there are 200 account codes a matrix record (referred to as a 'vector') would be 200 words long. Typical vectors in the matrix file are

   Debit vector (this month)
   Credit vector (this month)
   Year to date balances
   Year to date budget
   Last year - year to date (by month).

It is intended in the future to extend the matrix to include projected expenditure, income, etc., details so that financial modelling can be performed merely by specifying a scaling factor for groups of accounts. Thus expense items could be increased 10 percent, salary items 14 percent and sales say 16 percent and a series of financial statements run to access the effect of the various factors.

The above arrangement of files enables financial transactions to be posted to the matrix (i.e., ledger) file without prior sorting and coded either by account code (as in a conventional system) or by account mnemonics. It is on the reporting side, however, that the relational approach really comes into its own. Not only is the organisation of the files amenable to on-line account enquiry status, or to financial modelling, but because all the relationships in the accounting system are specified the system can derive a value for any accounting entity in the system. Thus cost of sales could be immediately derived for any of a number of trading departments. Gross profit for the enterprise can be derived by decomposing the structure below the level of gross profit, extracting the basic accounting information from the matrix file, and reassembling the results according to the relationships specified for the structure. This phase is currently under development.

Also under development is the last phase of the system, the report generator aspect. This is similar to those described for data dictionary systems. Report variables are again referred to by mnemonic name, thus easing user communication problems with the system. The system first looks up the data dictionary and if the name refers to an elementary item (one for which an account code is given) the system will go directly to the matrix file, read the appropriate vector if it is not already in core, and using the relative location stored in the cross-reference file, read directly the value for the particular account, supplying either the item description stored in the dictionary file or a user supplied description. If the name is not an elementary item the system will go directly to the relational file, decompose the structure, and derive the appropriate value at the level desired. Again this approach is adaptable to on-line or batch reporting and by specifying the appropriate vector number projected results can be derived as easily as current results.

6. SUMMARY

The first part of this paper examines the concept of system efficiency, both from the point of view of the machine and from that of the user. The assumption is that if the user, whether he be a designer or an actual user, finds a system to be efficient, then communication with that machine is improved. In particular, efficiency has been improved through the development of data base systems, which through the use of data structures enable an organisation to more accurately model its data resources. The development of the relational model of data provides an even more effective way of representing that data and the various interactions that exist. With the relational approach all files are viewed as flat and the opportunity to develop a data algebra has potential in both the hardware and software sectors. The introduction of the data dictionary further exhances the user-system interface.

In his studies of the relational model, and of the possibility of adapting accounting systems to mini-computers the writer was struck with the fact that an accounting system could be represented as a relational model. This approach, when combined with a data dictionary, could lead to substantial improvements in user efficiency. In fact, in the technique described in this paper, the actual files were possibly not as efficient as with conventional systems as a certain amount of redundancy was introduced. On the other hand user communications is improved through the use of mnemonics and with the relationships defined in a 'relations' file, logic, normally embedded in a user program, is held externally, thus increasing system flexibility. The system is therefore readily adaptable to the needs of different firms. The use of defined relationships enable derived data to be readily extracted using a decomposition process, while the use of a matrix file offers the potential for financial modelling with a minimum of additional effort. Overall the use of data relationships particularly as applied to an accounting system, appears to offer significant advantages, especially to the user, over conventional systems. It seems quite possible also that the approach could be applied to other commercial systems.

6. REFERENCES

1. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Communications ACM, v13, n6, pp. 377-387 (1970).
2. Chamberlain, Donald D.: Relational Data-Base Management Systems. Computing Surveys, v8, n1, pp 43-66 (1976).
3. Schussel, George: The Role of the Data Dictionary. Datamation (June 1977), pp 129-142.

IMPROVING COMMUNICATION BETWEEN THE ANALYST AND END USER IN THE SYSTEMS DESIGN PROCESS

AUTHOR: B.G. JENKIN BE, BSc
Advisory Systems Engineer,
Data Centre Services
IBM New Zealand Limited.

ABSTRACT:

A basic objective of the systems design process is to develop a system which satisfies the user's expectations of it. Accepting that the usual methods of communication between two or more people are potentially subject to both 'errors' and 'noise', as well as subjective filtering of the information, then difficulties in attaining this object are introduced at an early stage. Methods of improving this portion of the process are available.

Various techniques can be used to ensure that both parties are referring to the same concept. These can reduce some of the communication problems, and ensure a more satisfactory end product. This paper considers a number of the causes of inherent difficulties in people-people communication, and indicates how use of the established programming techniques of STRUCTURED DESIGN, PSUEDO-CODE, and WALKTHROUGHS may be used to resolve these problems.

KEY WORDS:

Systems design, Shannon Communication model, systems documentation, flowcharting, structure diagram, structured design, pseudo-code, walkthroughs.

C.R. CATEGORIES :

3.50, 4.19, 4.29, 4.43, 4.49, 4.6

1. OBJECTIVES

To consider systems design from a people point of view, indicating some of the problems in people-people communication, and how the appropriate documentation philosophy may assist in resolving problems. The subject will be considered under the following headings:

(a) General aspects of the systems design process
(b) Inherent difficulties of people-people communication
(c) Requirements of systems documentation
(d) Consideration of methodologies
(e) Implementation of new techniques
(f) Conclusions

In the discussion following, the term end user (or user) refers to the broad grouping of all those in the user department, excluding specialist systems staff, but including those responsible for initiating and stating the system requirements, as well as those who are required to use the system on a day by day basis.

The implementation of any software system involves a number of distinct stages. In general these stages are reasonably well defined, and separated insofar that checkpoints are usually established, relating to the completion of a defined portion of the work.

These phases may be considered as follows:

Requirements and objectives
System design
Programming and testing
Handover

This paper is only concerned with the systems design area, up to the provision of a systems/programming specification.

Many systems designers have run into the problems of misinterpretation of information given or received. The problems are well known, resulting in user inconvenience at the best, to catastrophe. Methods have been developed and are in use to improve this aspect of the process. However, I believe that the combination of pseudo-code and formal walkthroughs offer the best solution, since the techniques are simple to learn, and the benefits readily apparent. On the basis of personal experience, I would recommend their general introduction and use.

2. GENERAL ASPECTS OF THE SYSTEMS DESIGN PROCESS

Systems design is usually initiated as the result of a user need, stated as a set of requirements and/or objectives, in most cases specified by the users. If well defined, then full details of all inputs, all required outputs, and essential calculations may be given, but with no assumptions made as to the mechanism of internal transformation. The user makes the prime assumption, without necessarily stating it, that the internal operations of the transformation conform to his expectations, and not some alternative translation of them.

It is the job of the designer to create, from this starting point, a coherent documented system design, which will satisfy the user's expectation, and provide a definitive document for programming. Considering this phase from the two viewpoints, user and designer, the following genral objectives can be stated (not necessarily in order of importance) for each:

USER'S REQUIREMENTS AND OBJECTIVES

1. System which conforms to expectations, both in facilities and operation.
2. Realistic schedules in terms of the system concepts.
3. Consulted throughout development.
4. Vital changes can be accommodated.
5. System allows for growth.
6. Designer (and project team) visibly understands user needs.
7. Design documentation has sufficient detail to permit user to easily determine adequacy.

DESIGNER'S REQUIREMENTS AND OBJECTIVES

1. Design methodology that is easy to use and clearly indicates problem areas.
2. Standards that are fair but not restrictive.
3. Documentation that can easily be checked and approved by the user.
4. Minimisation of paperwork.
5. Provide a system that is easy to use and maintain, yet involves state of the art design.

6. Users who clearly state requirements.
7. Management involvement.
8. Control of user required changes.
9. Realistic schedules based on the system requirements.

The systems designer frequently faces a poorly defined work requirement, a well defined accountability system, and is always to blame.

3. PEOPLE-PEOPLE COMMUNICATION

One of the prime causes of software errors is mistakes in translating information, for example, whenever there is a failure to completely and accurately translate one form of the problem or solution to another more detailed representation. The human brain has the ability to match input against storage (filtering, inferring etc), seeking what is wanted or expected rather than what is there, making assumptions about missing facts, or overlooking information. If it is appreciated that data is missing, then the tendency is usually to interpret, sometimes incorrectly. Since few people write or otherwise express themselves clearly, the basic means of communication are inherently prone to error. The speed difference between thinking and writing (or talking) speeds can produce short cuts, increasing the probability that the next person in the process will make translation errors.

THE SHANNON COMMUNICATIONS MODEL

It is useful at this stage to consider the person-person communication link in terms of the Shannon Communications Model, as shown in Figure 1. Although this model was defined for electrical systems, it has valid application to this situation provided the definitions are changed. However, I use it as a descriptive, not mathematical model. In this context, consider each link of the model:

1. Message source - regarded as that portion of the brain which generates concepts, ideas etc.

2. Encoder - also the brain, but that portion converting concepts into an external form of speech or physical movement. In an electrical system this is usually regarded as an error free process, but in the human situation may be very error prone. At times it could almost be considered as a random generator followed by a set of filters, (these filters being created by experience, training, education, environment, etc) which modify and limit the conversion.

3. Channel - either as a piece of paper, or other physical media, or an air path in verbal communication. Both are subject to 'noise' and other interfering conditions.

4. Decoder - again the brain (of the receiving person), and subject to the same filtering action.

'Matters are often made more and more complex by the ability of man to play elaborate games that feed on themselves to create bewildering structures of immense intricacy, which obscure rather than reveal. The only thing that these structures do reveal, is that man has the ability and the compulsion to play these conceptual games' - E. de Bono

4. REQUIREMENTS OF SYSTEMS DOCUMENTATION

Based on the above model, it becomes possible then to state some basic requirements of system design philosophy and documentation in order to try and reduce potential error sources.

1. Minimise complexity to reduce translation errors.
2. Ensure greater precision during any translation that do occur.
3. Improve methods of communication of information.
4. Provide a clear statement of a problem, understandable by all concerned, and with no interpretation required.
5. Detect and correct errors as early in the design phase as possible.

6. Provide a technique which enables successive refinement of the problem with minimal restatement at each stage to permit delaying of detail decisions until actually required. This may be stated as the Principle of Minimum Commitment:

   'At each stage in synthesizing the design, make no commitment beyond that necessary to solve the problem at hand'

Design documentation may be generally classified as prose, diagramatic or formalised design language. The diagramatic method is typified by flowcharts or structure diagrams, while for the latter, the requirements must be specified according to some formalised design language. Processors are usually required to analyse these statements of design and provide logical checking of the system.

It seems to me, that for each of the mechanical methods, the designer is required, sometimes at considerable effort, to learn a formal language in order to substantiate the design. The methods appear suitable for very large and complex systems, but excessively formalised for the typical commercial data processing systems that are found in New Zealand. It also seems that in many cases the formalised design language, while it may be rigourous in specification lacks the aspect of user friendliness and thus the opportunity for the user to be involved in the process.

5. CONSIDERATION OF SOLUTION METHODOLOGIES

STRUCTURED DESIGN

Structured design is the technique of starting with an outline of the highest level logic, and by subsequent refinement resolving the problem to sufficient level of detail. The principle of Minimum Commitment stated above applies during this process. The art of structuring does require greater forethought, but the uniformity of the technique contributes to error reduction, since refinement is taken at each point to the stage where problems are resolved. I would expect most designers to be already using this method of design, whether deliberately, or because it is their nature.

STRUCTURED WALKTHROUGH

Walkthroughs are a working tool to remove errors, ensure completeness, with emphasis on error detection not correction. Error correction or problem resolution is not carried out at the meeting, nor is design criticism.

It is an egoless quality assurance tool, and the more problems uncovered at a session, the fewer left for more costly stages of the development.

The reviewer walks through each portion of the design in a step by step manner. Attendees of the review can then assess the logic for errors both in concept and execution.

All errors and other points are noted and used as an action list to correct defects. An open and non-defensive attitude on the part of participants is essential.

The moderator of the meeting ensures that there is a smooth flow, and prevents lengthy discussions or digressions.

Adequate preparation for a walkthrough is critical, according to the following sequence:

1. Schedule review.
2. Prepare material for review.
3. Distribute, 2 or 3 days in advance.
4. Meet (maximum of 2 to 3 hours).
5. Resolve issues off-line.
6. If another meeting is required, because of the number of errors encountered, then it is scheduled.

DOCUMENTATION METHODS

The two techniques of structured design and walkthroughs are essentially documentation independent, but some documentation methods, particularly pseudo-code appears more suitable for implementing the philosophy. A number of methods are considered in subsequent sections.

1. Cause effect graphs - diagramatic.
2. Decision tables - diagramatic.
3. Pseudo-code - constrained prose.
4. Flowcharts - diagramatic.
5. Structure diagrams - diagramatic.

PSEUDO-CODE

Pseudo-code is ideally suited as the documentation method of the structured process. It is not an executable language and has no syntax requirement, but conventions should be adhered to. In the simplest form the three basic statement forms of 'Sequential', 'If-Then-Else', and 'Do-group', coupled with indentation of logically subordinate detail, provide sufficient constructions. Do-groups may take the forms of single loop, multiple loop, Do-until (condition is satisfied), Do-while (condition is satisfied). Additionally, the Case statement offers clarity in situations where a series of nested If-Then-Else statements would otherwise be used.

The most important point is that some convention must be followed, for readability and consistency.

The factor of simple control statements implies that users may also easily understand the logic, and assist in the detailed design approval and checking.

In addition, the pseudo-code can be further refined, if necessary, to provide the program design documentation. This has the advantage of removing one step in the sequence of translation processes.

Pseudo-code is programming language independent, but certain languages, for example, PL/I (and its dialects), Algol and Pascal cater for all the Pseudo-code constructions. Cobol, Fortran, Basic and Assembler can be written in a structured style, but with greater difficulty. Pre-processors are available in some instances to overcome these difficulties.

A brief example of pseudo-code is shown in Figure 2.

Because pseudo-code is readable by anyone, its use provides several major advantages:

1. Users can ensure that the pseudo-code matches their requirements.
2. Programmers can ensure that program code matches pseudo-code.
3. Programming language independent.
4. Since the design specification is useable for programming, a stage in the total process is removed. An alternative is to use analyst-programmers.
5. Constrains the design to a logical sequence.
6. Can be entered and updated using on-line program library maintenance systems.

Difficulties:

1. Constrains thinking to a logical format, which does not suit some people.
2. Objections from programmers of non-structured languages.

FLOWCHARTING

I consider that flowcharting has little to commend it in comparison to other methods. Any non-trivial application requires complex and detailed charts which are laborious to draw. In addition they are difficult to amend, and frequently are not updated to show system changes. Automatic flowcharting aids are available, but even then the method of formatting may leave something to be desired.

STRUCTURE DIAGRAMS

This method provides a better picture of the structure of a program or design. However I consider it to be effectively a dual of flowcharts, and similar general comments apply to drawing and amending.

DECISION TABLES

Decision tables are a powerful tool for concisely defining complex logical relationships, but would be used in association with other techniques. However they bear little similarity to the structure of high level programming languages into which they are translated. Automatic conversion programs are available to translate to high level code, but the code generated is frequently difficult to read.

Advantages:

1. Less danger of omitting a logical possibility.
2. Better communication between interested parties (provided they are willing to understand the method).
3. Easily constructed and adaptable.
4. Very compact documentation.
5. Direct conversion to program code if processor is available.

Disadvantages:

1. Overkill for simple problems.
2. Indicates nothing of sequence.
3. Many programmers do not like them.
4. Considerable experience is required to use the method for a complex design, and it is not suitable for a user interface.

CAUSE EFFECT GRAPHS

This is a method for transforming a natural language specification into a formal language specification for logical completeness checking. If used in conjunction with decision tables, may also assist in deriving system test cases.

After identifying all input conditions and output actions, logical relationships are formed between them. Basically the method is a diagramatic form of Boolean logic equations, and as such, is a useful technique for those familar with the concept.

6. IMPLEMENTATION

No new method is a universal panacea. However pseudo-code and walkthroughs offer a reasonable compromise between the new techniques to be learnt and the improvement to be gained.

Pseudo-code is deceptively simple in concept, and for that reason designers may have considered it too basic. Greater forethought is required to conform to the structure, and remove language dependencies, than immediately apparent.

A drop in productivity may be expected during the initial stages, but this will be well counterbalanced by later gains.

The person best qualified to introduce the technique is not necessarily the most competent designer, since they will already have a high productivity in existing methods. An intermediate level analyst, who will enthusiastically grasp any opportunity to easily improve productivity may well be the better choice. The techniques can then be introduced gradually, until it becomes the installation standard. It is likely that users, seeing the advantages from their point of view will start to demand its use, thus hastening the change-over.

7. CONCLUSIONS

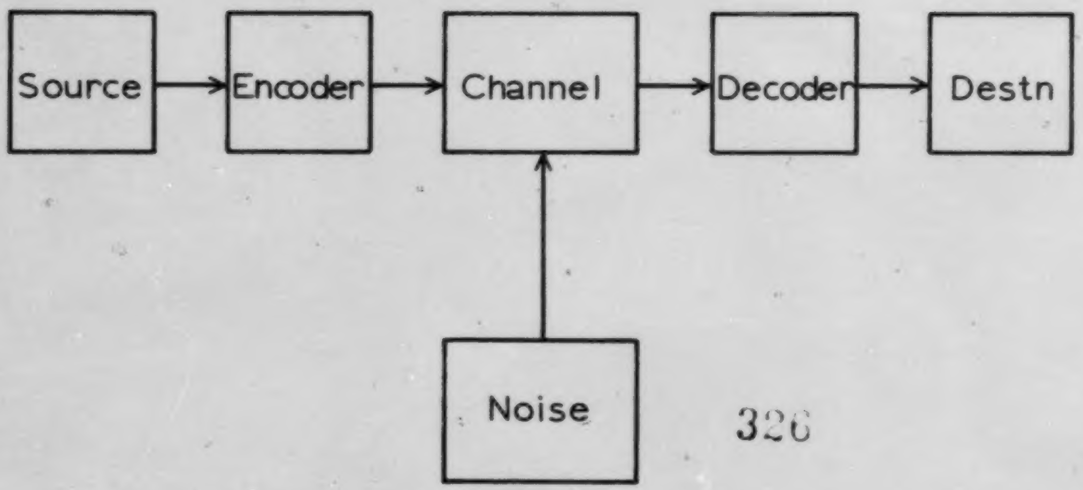
In this paper a number of design and documentation methods have been outlined. They all attempt to solve the basic problem of people-people communication, a process with error.

On the basis of ease of implementation, both by users and designer, I consider that pseudo-code provides, at the moment, the best vehicle of communication. Other methods do have a limited place, in a supporting role, but they possess restrictions which prevent satisfactory use in the user environment.

REFERENCES

1. Software Reliability - Principles and Practices, G.J. Myers, Wiley 1976
2. Computers in Business - an Introduction, D.H. Sanders, McGraw Hill 1975
3. Information, Computers and Systems Design, I.G. and M.E. Wilson, Wiley 1965
4. Information in Business and Administration Systems, R. Stamper, Batsford 1973
5. EDP Analyser, October 1972
6. Mechanism of the Mind, E. de Bono, Pelican 1971
7. Decision Tables: Theory and Practice, S.L. Pollack, H.T. Hicks and W.J. Harrison - Wiley Communigraph Series On Business Data Processing
8. The Politics of Software Systems Design, P.G.W. Keen and E.M. Gerson, Datamation November 1977
9. Comparing Software Design Methodologies, L.J. Peters and L.L. Trapp, Datamation November 1977
10. Higher Order Software - A Method for Defining Software, M. Hamilton and S. Zeldin, IEEE Transactions on Software Engineering Ma 1976
11. A Very High Level Programming Language for Data Processing Applications, M. Hammer et al, CACM November 1977
12. An Approach to Structured Programming, Parts I, II and III, R Doran and G. Tate, Massey University Computer Unit Publications 6, 9 and 10, 1972
13. An Introduction to Structured Programming in PL/I, IBM Manual GC20-1771

Figure 1.


Source
Encoder
Channel
Decoder
Destn
Noise

Shannon Communication Model

```
IF Issues-record
THENDO
    IF Issue-qty less than, or equal to, Balance-on-hand
    THENDO
        Unit-price = Value-on-hand / Balance-on-hand
        Value-of-issues = Unit-price * Issue-qty
        Balance-on-hand = Balance-on-hand - Issue-qty
        Value-on-hand = Value-on-hand - Value-of-issues
        IF Balance-on-hand = 0
        THENDO
            Add value-on-hand to Rounding-error-account
            Value-on-hand = 0
        ENDDO
    ENDDO
    ELSEDO
        Unit-price = Value-on-hand / Balance-on-hand
        Actual-issues = Balance-on-hand
        Value-of-issues = Unit-price * Actual-issues
        Balance-on-hand = 0
        Value-on-hand = Value-on-hand - Value-of-issues
        Add Value-on-hand to Rounding-error-account
        Value-on-hand = 0
        Issue-qty = Issue-qty - Actual-issues
        Create a Back-order-record for Issue-qty with this
        process-date
    ENDDO
    Update process controls
ENDDO
```

PORTION OF STOCK SYSTEM

Figure 2

SYSTEM CONTROL VIA A HIGH LEVEL LANGUAGE

Author: R.P. Lawson B. Sc. (Hons)
Senior Systems Analyst,
International Computers (New Zealand) Limited,
Wellington.

The most common means of communication between man and computer is via a control language. Unlike the common high level languages, no international standard for a control language has evolved, and on most systems the language is built round the operating system's requirements, often with prejudice to its ease of use.

SCL (System Control Language) is the control language used on ICL's larger 2900 series machines using VME/B. Possessing the more familiar properties of a high level language - particularly the separation of the grammar from the vocabulary - it supports a complete range of machine control requirements ranging from overall system control by an operator to submission and control of an individual user's job.

This paper discusses the advantages of the 'high level' approach, and the basic concepts in the design of SCL, and briefly describes some of the language's facilities.

KEYWORDS: SCL, System Control, Job Control, High Level Language.

C.R. CATEGORIES

4.22, 4.32, 4.35, 4.39

SYSTEM CONTROL VIA A HIGH LEVEL LANGUAGE

The designer of an application package - payroll, stock control or whatever-has at his disposal numerous programming languages such as COBOL, FORTRAN, PL/1, which have achieved a wide range of acceptance and are subject to international standards. When one comes however to the job control language necessary to develop and run these packages no such standards exist and there are almost as many JCL's as there are different operating systems, most of them reflecting in some way particular features of that system. Moreover, it is possible that within one operating system the JCL for batch work and interactive terminal work will be different, and likely that the operators of that system will be using yet another different control language. Yet a written control language is the most common means of human communications with the computer (if we except voice communication, light pens etc) and despite the differences in the languages we should be able to establish certain features common to all JCL's. These fall under one broad heading - communicating one's requirements to the system - and, I suggest, may be summarised as follows.

- A statement of resources required, e.g. files, tape volumes.
- A means of invoking application packages, compilers, sorts etc.
- A means of presenting data - for example, numeric data to be processed by a statistics suite, or source statements to be compiled by a FORTRAN compiler.
- Some measure of control over 'steps' within the job - i.e. performing subsequent steps only after appraising the results of an earlier step.

- Some measure of control over error conditions.

Most of these will apply to both batch and interactive terminal work. Similarly, if we look at the operator's task, he is effectively submitting one big 'job' i.e. overall control of the system - controlling (at some level) the amount of work in the machine, the various peripherals and possibly a communications network.

In this paper I should like to discuss a particular control language, SCL; I will deal with its overall design aims, and give a brief description of the language and facilities which have emerged to satisfy those aims; there follows a brief discussion of the implementation of the language, the execution image of the language thus attained and the possible effects of a different type of implementation.

DESIGN AIMS OF SCL

Two types of control language have influenced the aims in terms of what to avoid and what to try to achieve. The first is typified by the OS and System 4 JCL's, which impose a very rigid syntax of almost hieroglyphical characters which are very heavily oriented to their operating systems. The user must learn, for each facility he wishes to use, the exact syntax of each JCL statement involved. An opposite approach was taken by the Manchester University Department of Computer Sciences for the MU5 system, wherein the JCL consists basically of a call on some high level language program, which itself calls various system-supplied procedures to obtain resources etc. In this case the JCL is effectively the FORTRAN, COBOL (or whatever) program itself. The approach is somewhat inflexible, depending as it does on the capabilities of the particular language, but it does demonstrate two important principles - the use of a genuine high level language for job control and the provision by the operating system of a procedural interface for resource allocation and control. These principles are fundamental to the design of SCL, the aims of which may be summarised as follows.

1. SCL as a high level language -
   its grammar and syntax do not demand of the user continuous awareness of the idiosyncrasies of the hardware or operating system. The more important properties of such a language I consider to be as follows :
   - the separation of grammar from vocabulary. For example in FORTRAN terms, an IF statement is a part of the formal grammar while a named function subroutine is part of the vocabulary introduced by the programmer. Thus even fundamental operations - JOB, LIST, NEWFILE, COMPILE etc., are not built-in to SCL but are part of a readily extensible and replaceable vocabulary of commands. This means that users or installations can provide their own version of a command either by adding it to the vocabulary e.g. JIMSNEWFILE, or by replacing the original NEWFILE with a 'local' one of the same name. Here SCL differs from the superficially similar GEORGE family of control languages to which the basic commands are built-in. The ability to extend and replace vocabulary will be dealt with in more detail; it implies another important high level language property, i.e.
   - the ability to collect together a set of commonly used job control statements into a single named object - an SCL macro or procedure, which can be preserved and subsequently executed. This makes the user's task of communicating with the machine simpler in that invocation of a single SCL macro may cause a whole series of basic functions to be executed, functions which otherwise would have to be submitted separately.

2. A common language for all users -

   The language used by an operator, or by a user submitting work in batch or interactive mode, or by an engineer running test programs, is the same language. The vocabulary may be different for the different types of user, but they may transfer roles without having to learn a completely new control language. In practical terms this is to most advantage for the end user whose work is split between batch and interactive modes of working, as a set of SCL statements will work without change in either mode.

3. A 'window' on the operating system -

   All the basic system functions e.g. job submission, file creation and assignment etc., exist as procedural interfaces which are visible at the user level. They act as a set of 'windows' through which the operating system may be seen. While the typical user will rarely resort to direct use of these basic interfaces (for reasons to be discussed) they act as a series of building blocks which, allied to the SCL macro/procedure facility, allow for very powerful facilities to be created.

The net effect of achieving these aims is a language in its own right which has a high degree of machine independence, and enables the user to define his own job control facilities which may be as simple or complex as he requires.

FACILITIES PROVIDED BY SCL

This section is intended to give a flavour of the language, by illustrating its salient features with some bias towards those which, if not unique, are perhaps unusual in control languages.

1. SCL Variables, Constants, Expressions

SCL provides three basic types (or *modes*) of variable and constant, INT BOOL and STRING, with assignment statements and expressions to conduct computations and comparisons.

INT and BOOL variables hold simple numeric and Boolean (true/false) values respectively: for example,

```
INT OPEN_FILE_COUNT :=0
OPEN_FILE_COUNT :=OPEN_FILE_COUNT + 1
BOOL ALL-FILES-OPEN := OPEN-FILE-COUNT = 5
```

STRING variables hold character strings, and a set of concateration and removal facilities is provided, e.g.

```
STRING  DATE := "1978/MAY/10"
STRING MONTH := DATE AFTER"/" BEFORE"/"
BOOL OYSTERS := MONTH INCLUDES "R"
```

There are also SUPERSTRING variables which may hold lists of strings individually accessed via a subscript, e.g.

```
SUPERSTRING INFILES := 'MAY78" & "JUNE78" & "JULY78"
STRING THISFILE := INFILES(1)@ i.e. "JUNE78" @
```

(Note the use of @@ to delimit comments in the SCL text).

SCL variables are held in a location known as JOBSPACE, which is maintained by the SCL execution system. The latter also provides procedural interfaces via which the user may access JOBSPACE variables from other languages.

2. Control of Flow

In addition to the calling of an SCL procedure or macro, control may be transferred conditionally or unconditionally from one SCL statement to another, via use of IF and GOTO statements which behave in similar fashion to their counterparts in other high level languages.

For example:

```
GOTO LAB1
```

unconditionally transfers control to a statement identified by a label of the form

```
LAB1:
```

Simultaneously,

```
IF   ALL_FILES_OPEN
THEN  PROCESS_DATA
ELSE  GOTO OPEN_MORE_FILES
FI
```

tests the ALL-FILES-OPEN condition; if true, a program called PROCESS-DATA is executed, otherwise control is transferred to another statement labelled OPEN-MORE-FILES

There is another type of conditional, the panoramic conditional, which nominates some action to be taken whenever a particular condition arises, e.g.

```
WHENEVER ERRORCODE = 6 THEN GOTO ERRORPATH FI
```

This statement declares an interest in the variable ERRORCODE, such that whenever it receives a new value this value is checked and if it happens to be 6, then control is transferred to the SCL statement identified by the label ERRORPATH.

This panoramic conditional takes effect as soon as it has been encountered in the SCL source, and avoids the need to write explicit checks on the value of ERRORCODE following each statement in which it is assigned a new value. Thus the user has the ability to make run time decisions, to determine which step in his job will next be executed and under which conditions. The IF and GOTO statements are obviously important in batch job control programs, but not relevant to interactive work; the WHENEVER statement is useful in both cases - in interactive work it could be used, for example, to generate a comprehensive error message whenever a program returns an error code;

```
WHENEVER ERRORCODE NE 0 THEN OUTPUT_ERROR_MESSAGE FI
```

where OUTPUT_ERROR_MESSAGE could be a user written SCL macro or procedure to produce this message.

3. Block Structure

BEGIN and END statements are used to create and end new lexical levels which determine the scope of SCL variables, in a similar manner to the block structure in ALGOL. Under VME/B the block structure has a more wide ranging effect - it is reflected not only in SCL, but also in the part of the operating system which controls resources - files, volumes etc - such that a job's allocation of resources is determined by its block structure. The scope of a resource is defined as the block in which it is allocated. In addition the block structure also determines the extent to which a program is allocated store upon being loaded - at the end of the block in which it was loaded, the store is released and the program effectively 'unloaded'. As it is block end which determines unloading, several programs may be loaded simultaneously.

By judicious use of block structure therefore the VME/B user is enabled to exercise tight control over his resource allocation and store requirements.

4. SCL Macros and Procedures

SCL statements may besubmitted 'in line' as a batch control program or interactively from a terminal or operating console. In these cases they are said to come from the job control stream. Alternatively they may be grouped together within named SCL Macros or Procedures, which may be parameterised.

As currently implemented SCL Macros and Procedures are treated in much the same manner; entry to either causes execution to switch from the current control stream to the statements of the macro or procedure which therefore became a sort of 'pseudo-control stream', and are executed until exhausted or otherwise terminated, at which point control returns to the original control stream.

Thus the SCL macro scheme is unlike some other macro schemes which involve substition of complete macro text at the point of invocation - the effect in SCL is more akin to entry to a procedure.

The principal difference between an SCL macro and procedure is that whereas the procedure introduces a new lexical block by executing a BEGIN, such that all variables, resources etc., declared or allocated by the procedure disappear upon its exist, the macro has no such effect. Instead, an entirely independent lexical level is created, called MACSPACE, in which SCL variables (which otherwise have identical modes and properties to JOBSPACE variables) may be declared. Variables are identified as MACSPACE variables by being prefixed with'%'. Either MACSPACE or JOBSPACE variables may be declared within a macro, but whereas the MACSPACE variables disappear when the macro ends, any JOBSPACE variables declared will remain in existence. The MACSPACE lexical level is held completely separately from JOBSPACE, and has no effect other than to limit the scope of MACRO variables - i.e. entry to and exit from a macro has no effect on the system block structure which controls resource and store allocation. MACSPACE can therefore be considered as a working storage for the execution of the macro only, whereas any JOBSPACE variables declared or resources allocated will have residual effect.

An important facility, available only within SCL Macros, is textual substitution. A MACSPACE variable of mode STRING may appear anywhere in the

```
 MACRO COMPILECOBOL IS
   (LITERAL %SOURCEFILE,
    LITERAL %OBJECTFILE,
    LITERAL %OPTIONS := "NOMAP",
    INT     %ERRORLIMIT := 10,
    REF INT %RESULTCODE := RESULT)
 MACBEGIN     @ CREATES A NEW MACSPACE LEXICAL @
@LEVEL @
 @ BODY OF MACRO STATEMENTS @
 MACEND     @ DELIMITS MACRO BODY, ENDS @
@MACSPACE LEVEL  @
```

Looking at the formal declarer in more detail, we see that the macro is to be named COMPILECOBOL and that it will have five parameters. The parameters are named %SOURCEFILE, %OBJECTFILE etc., and their *keywords* are the names without the prefixing '%'. They will exist as MACSPACE variables while the macro is being executed, but note that we have here modes, namely LITERAL and REF INT, which were not mentioned in the previous section on SCL variables. These modes only apply to parameters; a LITERAL parameter is in all respects similar to a STRING variable, except that the enclosing quotes are omitted from the value passed on a call of the macro. The REF INT mode allows parameters to be passed by *reference* rather than by *value*, such that they may be written back to. The last 3 parameters are given default values; for the RESULTCODE parameter this means that by default a reference is set up (on entry to the macro) to an INT variable called RESULT.

The keywords allow parameters to be supplied in any order, thus reducing the chance of making error, and only sufficient characters of the keyword to identify it uniquely from other keywords of the same macro need by supplied. As a convention, all the standard system supplied macro keywords can be uniquely identified by their first three characters. Thus a typical call may look like:

```
COMPILECOBOL (SOURCE = PAYROLLSOURCE, OBJ =
              PAYROLLOBJECT)
@ defaulting OPTIONS = NOMAP, ERRORLIMIT = 10, @
@ RESULTCODE = RESULT @
```

Note the way in which SOURCE and OBJ parameter values are supplied as string literals without the quotes.

```
COMPILECOBOL (OPT = MAP, ERR = 5, SOURCE = XSOURCE,
              OBJ = XOBJECT)  @ overide OPT and ERR @
            @ defaults @
```

Note the random order allowed by the use of keywords. In interactive SCL further use is made of keywords in that the caller may invoke keyword prompting by not completing the parameter list - i.e. by omitting the ")". For example (the prompt is shown underlined, end of message as □):

```
COMPILECOBOL (□
*SOURCEFILE = XSOURCE,□
*OBJECTFILE = XOBJECT,□
OPTIONS = )□
```

macro text preceded by an extra "%", in which case the current value of the string variable is substituted into the source prior to analysis and execution of that source. For example, the following macro text;

```
.
.
.
STRING %OPTION  @ Declare a MACSPACE variable @
.
.
.
%OPTION := "MAP" @ Assume one of several points at @
 @ which 'option' is set @
.
.
.
COMPILECOBOL (OPTIONS = %%OPTION,.....)
```

Before execution, the latter statement becomes

```
COMPILECOBOL (OPTIONS = MAP,.....)
```

Note that the text substituted depends on the value of the variable at the time the statement is executed, <u>not</u> its value upon entry to the macro. The purposes to which textual substitution can be put are many and varied - for example complete SCL statements may be constructed at run time prior to execution.

The parameter interface to a Macro or Procedure allows for <u>keywords</u> and <u>defaults</u>. When a macro or procedure is declared to the system it is headed by a formal declarer which gives the <u>name</u> of the macro, and for each parameter a <u>mode</u>, <u>keyword</u> and (optionally) a <u>default value</u>. For example :

Execution of SCL

The foregoing description of some of SCL's facilities is only part of a picture which is incomplete unless some idea is given of how the SCL execution system works, and of the 'execution image' of SCL.

As implemented under VME/B, SCL is executed via an interpreter on a line-by-line basis. Thus SCL execution has an incremental effect on the state of the job, with respect to the number of resources so far allocated, SCL variables declared, programs loaded etc. This is true currently also of SCL statements within SCL macros and procedures. It is fundamental therefore that each SCL statement is executed within a context determined by all the SCL statements which have been executed before it.

It is appreciated that interpretive systems are not as efficient as compiling systems (because each time a line of source is executed it requires to be fully analysed as though never seen before). However, a recent exercise in attempting the design of an SCL compiler has shown that full compilation is only practically possible given a set of constraints and new language facilities which, in changing the nature of our current interpretive SCL, cannot be said to enhance its image. Several areas in particular proved difficult to resolve within a full compilation scheme.

1. Textual Substition

   As we have seen, textual substitution is an extremely powerful facility which depends on the run time value of the string variable to be substituted. There is no way for a compiler to predict what this value will be; so the compiler would have to be re-entered at run time to compile a single SCL statement - any which contains a textual substitution. Given the heavy use of textual substitution which we have seen in practice, it is evident that any advantages gained by pre-compilation would be lost by constant need to re-enter the compiler. It is also such a popular facility that mere suggestions of its possible removal (in the interests of efficiency) have been greeted with howls of dismay.

2. <u>Data embedded in the control program</u>

It is possible to present "alien data" - i.e. lines which are not SCL - within the control program. These would be (typically) COBOL source statements for input to a COBOL compiler, numeric data for a statistics program etc. e.g.

```
STATSPROG  (INPUT = CONTROLSTREAM)
  1010
  1012
   998
  1028
++++
```

Here STATSPROG is the program which is going to read its input data from the control stream, and will receive the 'end of file' condition when the standard <u>enddelimiter</u> - ++++ - is encountered.

Currently, it is not compulsory to use a "start delimiter" to mark the start of the alien data (i.e. immediately following the STATSPROG statement). Such a start delimiter would need to be compulsory if the job control program were to be fully compiled, otherwise the compiler would attempt to 'compile' the alien data. An alternative, compiler recognition of every command's potential use of the control stream for input, would be rather cumbersome, and would in any case conflict with the principle of separating grammar from vocabulary. It may not seem much of a constraint to apply the start delinator, but it is a constraint nonetheless which is absent under the current scheme.

3. Context Problems

There is a general class or problem, which we can call 'context problems' which are rather more serious than those of textual substitution and alien data.

Under VME/B, all programs, system or user, are organised into libraries, and a central piece of software, the program loader, is responsible for searching these libraries for named objects. Thus when the name STATSPROG is encountered in an SCL statement, the SCL interpreter, not recognising it is an SCL variable, invokes the program loader to find and load it. In order to reduce the amount of library searching the user has control over which procedures, and a simplified template, giving parameter modes only, is generated separately for them. This means that all parameters have to be supplied (there being no default) and in the correct order. When called from SCL, the SCL execution system uses the template to match the called object's parameter requirements against those provided, and rejects incorrect modes, excessive numbers of parameters etc. Because this interface is not as convenient as the keyword/default image supplied by SCL macros, compilers, utilities and the more commonly used system interfaces are enveloped in a short SCL macro which effectively provides the keyword/default image for them; for this reason the average user never uses the direct call on non-SCL objects, except of course for his own programs.

The next stage in SCL development will be to provide a mechanism for creation of keyword/default type templates as a separate operation if required, and to attach these to non-SCL obects. In this way it will be possible to call objects written in any language from SCL using the full power of the keyword/default interface. As a separate development, it will also be possible to call SCL objects directly from other high level languages which are capable of providing correct parameter mappings, thus introducing the possibility of true mixed language programming.

In this sequence the mandatory parameters - those for which there is no default - are indicated by a keyword prefixed by "*"; submitting a 'null' value by providing the parameter list terminator ")" indicates that the default is to be used, in this case for OPTIONS. The remaining 2 parameters are of course also given their default values. By continuously invoking keyword prompting the user will force a 'wraparound' and repeat prompts, to which he may supply new values if required. This keyword prompting facility is built into the system, i.e. the user only needs to specify the keywords required on his macro or procedure declaration, and does not need to code in the prompt (as in GEORGE3 for example).

The information on parameter keywords, modes and default values is held in a special object known as a template which is derived from the formal declarer and held with the body of the macro or procedure. The SCL execution system analyses parameter input against the template, and uses the template to drive keyword prompting.

5. Calling Non-SCL Objects

It is also a requirement to be able to call non-SCL objects from SCL, for example the operating system's inteface procedures, compilers, SORT'S, user programs etc. In practice the standard SCL macros provided by ICL each call one or more of the basic interfaces. In the present state of development, none of the compilers for the languages in which these are written generate the type of template seen on SCL macros and libraries are in context at any particular moment, via an SCL command EXTEND_CONTEXT. This will add another library to the 'top' of the existing context; and the program loader when invoked will search from the top of the list down until the required object is found. As an example, consider two libraries, L1B1 and L1B2. L1B1 contains several programs including PROGA. L1B2 contains another program named PROGA (which we will designate PROGA').

Assuming the user has called
EXTENDCONTEXT (L1B1)
to put L1B1 in context, a statement of the form PROGA
will cause the PROGA in L1B1 to be found, loaded and executed.

However, another context change,
EXTENDCONTEXT (L1B2)
will add L1B2 to the context and a subsequent call of PROGA will this time pick up PROGA'. This relies of course on the current state of the context being taken into account each time an SCL statement is executed.

Thus, if we try to compile an SCL procedure which contains EXTENDCONTEXT commands, and try to take advantage of full loader capabilities to fix up all references to called commands or programs prior to execution of the procedure, the context changes will effectively be ignored at run time, because the references satisfied on entry to the compiled SCL procedure will be those currently in the context at entry time, not those required by context changes programmed into the SCL.

Another problem in compiling calls to other objects - be they SCL or whatever - is that their parameter modes need to be known in order to generate the correct object code for the call. In interpretive SCL this is no problem, because, as we have seen, each callable object may hold with itself a 'template' which defines its parameters, and which is used by the SCL interpreter to match supplied parameters with those required. For compiled SCL it is possible of course to 'load' the templates of called objects at compile time, and for commonly used objects this presents few problems and few overheads. For less common objects however, which are not in the context at compile time (and whose templates are not therefore available) or which perhaps do not yet exist, a special type of SCL statement is required which can describe an interface in terms of parameter modes etc.

Another problem related to context is that of SCL variables referred to within some compiled SCL object but which exist outside it. In interpretive SCL a search is made of the job's SCL variables each time a name is encountered in the SCL source; if the name is not found it is assumed to be some callable object and the program loader actioned to find it. In compiling SCL however, the SCL variable search must be confined to the limits of the object being compiled (as those outside will not necessarily be the ones present when the object is executed); therefore any external variables to be accessed must be pre-declared with their modes to the SCL compiler via some form of EXTERNAL statement, in similar manner to unknown objects to be called.

It will be seen then that in order to compile SCL certain constraints must be observed, and that the 'execution image' of SCL must be quite different from that of interpretive SCL, in that it is no longer incremental to the same degree.

So what of the possibility of SCL compilation ? The constraints - compulsory start delimiters, heavy use of EXTERNAL statements - were found to be unacceptable, and in any case some form of interpretive (or more importantly, incremental) SCL is necessary for interactive or operator work. It is also desirable that an identical set of SCL statements presented as a batch job or interactively should have the same execution effect, which would not necessarily be possible (because of the factors just discussed) if the one were compiled and the other interpreted.

Although the efficiency of compiled SCL is estimated to be an order of magnitude greater than that of interpreted SCL, the overall system overhead of executing SCL is on average only a few percent of that incurred by the job as a whole.

The future course will be therefore to retain interpretation for SCL within the job control stream or macros; and to compile SCL procedures, whose language will be subject to the constraints discussed above. Therefore, a small but significant gain in efficiency can be made by writing SCL procedures, but it is for the user to trade this against some loss in useability.

ADVANTAGES OF THE SCL APPROACH

It was not my intention to conduct a 'flagwaving' session on behalf of SCL as the 'Grand Universal Control Language". Indeed, as we have seen, despite its high level nature several features relate directly to the way the underlying VME/B operating system works - for instance, implications of block structure and context. And even if a standard syntax and grammar could be agreed upon the vocabularies of different operating systems will always reflect the different ways in which they work, although (at a cost) the vocabulary of one system could be forced to 'simulate' another. Nonetheless these are two features of SCL which to my mind could be used to advantage in most control languages. The first is separation of grammar and vocabulary: this avoids the need to build any detailed knowledge of sytem functions into the control language. It requires some means of easily extending the vocabulary, but given that, the power and flexibility of the control language can be vastly increased.

Second, the keyword/default type of interface on commands, which serves the allied purposes of building into "the system" a mass of commonly used information and thereby reducing the amount of effort on the user's past, and also frees the user from slavish observance of detailed rules on position of commas, spaces etc.

So far as the SCL user is concerned there are several main advantages.

1. There is only the one language to learn for all types of machine usage:
2. He need only master the basics of the language (how to issue commands using the keywords/default interface) in order to be able to use the system, leaving his major learning effort for the facilities of the system itself.
3. New commands may be constructed using the standard issues as building blocks, allowing the user to define interfaces which suit his own requirements.
4. The capability is present for tight control of progress within the job and any errors which may occur, i.e. a high degree of intelligence may be built into the control program.

Most of these advantages arise because the job control language has been conceived as a genuine High Level Language in its own right, rather than a series of system dependant key combinations. It has proved a worthy approach to JCL, whatever the future may hold for a standard.

COMMUNICATION - FROM SYSTEMS REQUIREMENTS TO SYSTEMS ARCHITECTURE

AUTHOR: J.A. LENS VAN RIJN

ABSTRACT:

This paper identifies a need for improved formalised communication between the user of an automated information system and the various groups of people involved in its design and implementation.

It proposes a methodology for improving this communication and to convey a broad understanding of some of the techniques involved in this approach.

The scope comprises the full range of communciation required, from descriptions of existing system flows through specification of the desired information system. Some coverage is given to the specification of the technical system.

KEYWORDS AND PHRASES:

Systems Requirements, Systems Architecture, User, Communication, technicians, terms and symbols, documentation technique, object system, information system, technical system, Information analysis.

C.R. CATEGORIES:

2.4, 4.0, 6.0

1.0 THE NEED FOR IMPROVED COMMUNICATIONS

There is a growing awareness in the data processing community that, over the years, a quite startingly high proportion of systems have failed.

Surveys have been done which indicated that two thirds of the Users were dissatisfied with the results achieved.

Even in those areas where reasonable successes have been achieved, there is often a feeling of disquite regarding the apparently exponential growth in maintenance requirements for the systems developed.

It is possible to do an analysis of these failures which would indicate that they are caused by failures in communication:

a) From the users to the technicians regarding the complexities of their information requirements.
b) From the technicians to the users regarding the operational impact of the technical complexities incorporated in the system.

An illuminating comparison can be made with a well-established industry such as the building industry and the communication which is practised in this field.

It turns out that sketch plans and working drawings are an effective means of communication between all the parties involved. What is equally important is they make it relatively easy for the user to become involved in the design process.

Computer systems designers at an early stage in the design process introduce terminology and symbols (such as disks, tapes) which represent objects that are unrelated to the reality which the user himself is concerned with.

Inevitably, the point is reached where the client cannot relate what is being produced, to what he is trying to achieve.

The full realisation generally does not usually occur until the system is actually programmed, by which time it may be too late to respond to his concerns.

It is to be argued here, that a solution lies in :

a) In making a clear distinction between the specification of the information system, and that of the technical system.

b) In using a precise, structural and problem-oriented documentation technique for the specification of the information system.

2.0 THE CONCEPTUAL FRAMEWORK

The conceptual framework used in the approach under discussion makes use of the early ideas of Professor Borje Langefors of the University of Stockholm and much work that has gone on since these were formulated.

Three categories of system are recognised:

a) The object system: The system of real-world objects. The scope of the object system is such as may be relevant to the information system under consideration.

To be considered are sets of objects, their relationships, their typical properties.

The information system, to be considered next, is in a real sense a part of the object system which it is used to control. The control processes which link the information system into the object systems can well be studied in the context of the object system.

Making some assumptions about the object system, it may also be studied in isolation.

b) The information system: The lowest level of this is the elementary message which informs about a property for a particular object at a particular point in time.

A sales item, for example, may be shown on an invoice as being sold in a particular quantity at the time of the invoice.

The storage of the data (in this case as printed symbols on a piece of paper) is regarded as being entirely separate from (and largely irrelevant to) the study of the information itself. A data-element is not the same thing as an information element and the relationship between them need not be one to one. The representation of the information in terms of stored data is indeed a technical design choice.

c) The Technical System: It can sometimes be useful to make a distinction between the technical system which is developed in response to an information system specification and the way the component parts are linked together to form the run-time system.

Within the conceptual framework outlined above, the task of designing and realising an information system can be decomposed into three entirely separate activities.

a) The analysis of the object system with a view to establishing and designing the required control processes.

b) The analysis and design of the information system with as end-result, a precise and sufficiently detailed model of the required system (this can in fact be presented as a matrix).

c) The design and further analysis of the technical system with the objective of achieving the most efficient mapping of the required information structure on to the chosen technical equipment.

In the process of evaluating alternative technical structures and their efficiency, further technical data elements (such as pointers, label-records) will be added to support the technical solution envisaged.

As a result of the technical systems design the modules which are produced are linked into run-time job-streams. It is possible to consider this as a separate activity.

3.0 METHODOLOGY AND SPECIFIC TOOLS

The formalised approach, which is advocated as providing improved communication with the users and between the various groups involved, involves the following steps:

1) Current systems investigation. Alternatively, if the situation permits this, a full object system analysis may be carried out using a formalised documentation technique very similar to information analysis.

2) Information Analysis: This uses the basic technique of precedence analysis, with structuring according to control and logical (or information-technical) criteria.

3) Technical design and development planning. Because of the structural nature of information analysis it is possible to carry out this activity for any level of detailment.

   For costing and budgetary purposes it may indeed be desirable to perform this activity repeatedly for several levels of information analysis.

4) During the technical design there must be a strictly controlled process of feed-back so that all changes are reflected in the information - analysis document. A system of cross-referencing by means of matrices is used to relate the two designs to one another.

3.1 Object System Analysis

This activity may be carried out for one of two reasons:

a) To investigate the desired object system, with a view to designing the required control processes, as a result of the insights gained.

b) To carry out an investigation with systems boundaries which are defined as being wider than those of a particular information system.

The objective in this case would be to evaluate alternative conceptual structures for object categories. The information requirements defined together with the alternative forms of structuring these can then be input to the activity of database design.

Since the formalised approach to object systems analysis is not an indispensable part of the approach outlined here, it will not be discussed in further detail.

3.2 Information Analysis

3.2.1 Introduction

The approach to information analysis consists of defining (in a structured way) subsystems of information sets and their precedence relationships.

These subsystems are in their turn further defined in exactly the same manner, until the required elementary level is reached.

In practice, the minimum elementary level is where document and record layouts can be referenced.

The precedence relationship referred to above determines which sets must logically be present for it to be possible to produce the required output sets.

3.2.2 Brief Example

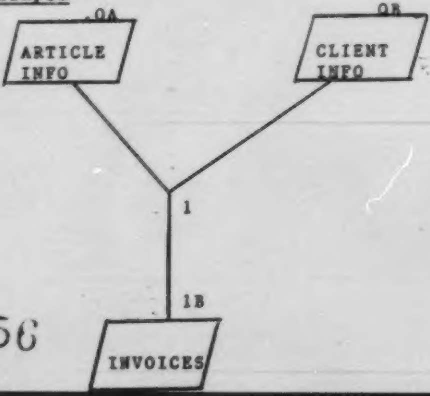


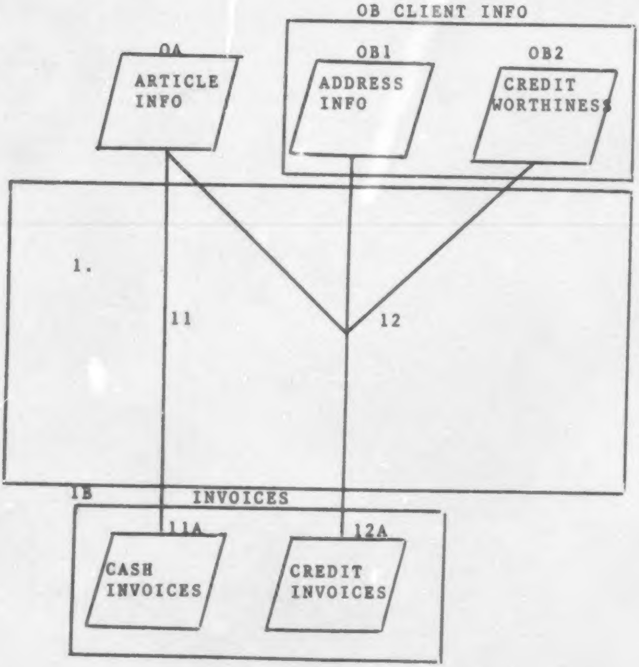
FIGURE I

The precedence relation depicted in fig. 1 indicates that in order to produce set IB INVOICES it is necessary to be able to access the sets OA ARTICLE INFORMATION and OB CLIENT INFORMATION.

The relationship defines a process labelled 1, in which the production of invoices occurs.

Assuming that the elementary level has not yet been reached, process 1 will be further detailed in a separate graph.

FIGURE II


OB CLIENT INFO
OA
ARTICLE INFO
OB1
ADDRESS INFO
OB2
CREDIT WORTHINESS
1.
11
12
IB
INVOICES
11A
12A
CASH INVOICES
CREDIT INVOICES

In fig. 2 set 1B is seen to decompose into two output sets which are always given numbers relating to the process from which they issue.

Set OB is found to decompose into set OB1 and set OB2.

For each process, the process logic is documented in the form of a decision table insofar as this is necessary.

On inspecting the process logic of process 12 it might be established that there is a need for a credit rule table, which it may be desirable to show as a separate set. In that event it will be necessary to alter graph 1, but also the level above (fig 1.), which depicts a portion of graph 0.

For each set the size and the number of occurrences are documented together with time frequencies.

If the number of occurrences for B1 and B2 are both equal to OB then obviously in technical design it will be possible to combine them into one record.

3.2.3 Checklist for Information Analysis

The following procedure is recommended for carrying out information analysis.

1) List input and output sets to define the system boundaries. Document any specific exclusions.
2) In working papers collect notes on composition of sets.
3) Make a large chart incorporating several levels of detail, on one sheet of paper.
4) Consider alternative process and set groupings using criteria, such as decision centres, developed during object system analysis. The process-matrix may be a useful tool.
5) Draw first and second level charts with set and process specifications. Detailing of set specs. occurs on the preceding level.

6) Discuss "hierarchically", initially omitting process details. Refine with the required amount of iteration, completing each level in full before the next level is tackled.

Points to Note:

1) Avoid computer terminology. Be meticulous with nomenclature.
2) Decomposition of sets must be based on critical analysis of information content.
3) Beware of building in design choices (Manual/computer process; real-time/batch).

3.3 Technical Design

The approach to technical systems design consists of a systematic evaluation of alternatives using the following as input.

- information analysis
- predefined performance objectives (e.g. average response time 3 secs.; maximum nightly batch run time 1 hour).

The evaluation will also use certain assumptions about hardware limitations such as numbers of available channels, memory size and strategies such as blocking factors and overlay techniques. These will be made explicit.

The evaluations are carried out in two main areas.

a) Process consolidation (to minimise data transport)
b) Record/file consolidations (to minimise storage, mainly by the sharing of fixed overheads).

The outputs of the activity are

a) Documentation of the alternatives considered, the results of the evaluation made and the choices arrived at.
b) Documentation of the mapping of the information system onto the technical system in the form of

matrices (Infoset-file matrix; process-module matrix).

As the design progresses, technical data-elements and technical processes are added, and are kept recognisable as such in the documentation.

At all stages of technical analysis, specific techniques (such as Warnier/structured program analysis; conversation flows) may be appropriate. It should always be possible to integrate these into the overall approach.

4.0 SKILLS REQUIRED FOR SUCCESSFUL APPLICATION

The skills which need to be conveyed to anyone wishing to put into practice the approach, which has so far been outlined in a very summarised form, are the following:

- Understand the basic tools
- Thorough appreciation of the conceptual framework
- Appreciation of practical short-cuts.

In teaching users the documentation techniques used there are certain errors which are commonly made which can be shown by practical examples to be in conflict with the basic objectives.

In teaching those with a computer systems background it is important to make continuous reference to the conceptual framework in practical examples. It may be a useful exercise to demonstrate how exposure to such project-control oriented concepts as modular programming can lead to the development of unsatisfactory structures.

Since the approach outlined can be very time-consuming because of its iterative nature, it is important to convey an appreciation of how the required results can be achieved within a reasonable timeframe without violating any of the basic concepts.

5.0 BENEFITS AND PROBLEM-AREAS

The main problem areas in the approach are the following:

a) The documentation technique is fairly laborious. Some computer aids have been developed, but are not generally available.

b) The understanding of the conceptual framework and how this actually works in practice cannot be conveyed in a short space of time. Initial enthusiasm can quickly turn into disappointment if further guidance is not given at the appropriate moment.

The benefits have already been outlined to a large extent. There are good grounds for the claim that the approach can overcome the communications gap which has been identified.

In addition, the method provides a fundamentaly systematic and theoretically sound approach to systems development using a far less intuitive basis than is generally encountered.

CENTRALISED COMMUNICATIONS FOR THE MELBOURNE FIRE BRIGADE

G.L. Mann BA(Hons), B Com, MNZCS;
Manager, Services Division

T.W. Marks BE(Hons), Ph D, MIEEE;
Manager, Control Systems Division

Public Management Services
C/- Systems & Programs (NZ) Ltd.,
Cnr Dudley and Margaret Streets,
LOWER HUTT

SUMMARY

The powerful aids that computer technology can provide to firefighting services are only slowly being recognised. This paper describes the impact that the new techniques are having on two major areas of the operations of the Melbourne Fire Brigade: an extensive telemetry system for alarm monitoring; and computer-aided despatch.

KEY WORDS

Computer-Aided Despatch, Alarm Monitoring, Alarm Interface Unit, Fire Station Equipment, Command and Control, Call Initiation, Call Location, Call Status, Appliance Assignment, Appliance Despatch, Appliance Status, Digital Radio.

C.R. CATEGORIES

3.81, 3.82, 3.83, 3.89, 6.9

1. INTRODUCTION:

The Melbourne Fire Brigade is one of the largest in the world. It concerns itself with the protection of an area of approximately 1000 km^2, which includes most of the Melbourne Metropolitan area. There are 48 fire stations, divided into five districts, each headed by a district station. Some 1450 men are employed on direct firefighting activities; they work a cycle of four days on, four days off. During their days on the men work two day-shifts of 10 hours followed by two night-shifts of 14 hours. There are currently 98 appliances in use at fire stations. The basic appliance is the hose carriage, of which there is at least one at each fire station. Other appliances, such as ladders, snorkels and salvage vans are located at strategic places throughout the area.

2. FIRE ALERTING SYSTEMS:

There are three major methods of notification of fire:

1. By telephone to the control centre at Eastern Hill (Melbourne City);
2. By telephone to a local fire station;
3. By means of the private alarm system, whereby approximately 5250 fire alarm systems in buildings and factories around the metropolitan area are connected by private leased lines to the nearest fire station. Alarm systems may be of the thermal type, sprinkler type, or the MFB's own break-glass, push-button type.

3. APPLIANCE DESPATCH - EXISTING SYSTEM:

3.1 DESPATCH FROM CONTROL CENTRE:

The despatch from the Control Centre occurs for all fire calls received there. The location of the call is ascertained and the Control Centre notifies the relevant district station on the direct lines of the internal telephone system.

The district station then relays the information to the appropriate local stations. All despatch details must be recorded at the Control Centre, district station and local station.

3.2 DESPATCH FROM LOCAL STATION:

If a call* is reported directly to a local fire station, the local station notifies its district station, which in turn notifies the other local station in the turnout area, and then the Control Centre.

3.3 PROBLEMS WITH THE EXISTING SYSTEM:

a) Age: The entire system operates quite effectively, but was new in 1914. Parts are becoming extremely difficult to replace.

b) Speed of Despatch: With up to three relayings and recordings of call details, valuable seconds may be lost in the despatch of appliances.

c) False Alarms: Some 32,000 calls are received by the MFB in a year. Of these, approximately 9,000 are for legitimate fires. The remaining 23,000 are false alarms and of these 60% are attributable to faults in the automatic alarm system.

d) Testing of subscriber alarm systems: The rule is that each thermal alarm system must be tested daily, each sprinkler system weekly, and each MFB push-button system monthly. All such testing must be monitored by MFB personnel at the local watchroom. Testing is accomplished by a person at the premises notifying by voice along the line that a test is to be made, making the test, and receiving voice communication that the test has been successful. The successful test is recorded and defaulting subscribers must be chased up.

Monitoring of these 5250 alarm terminations, both for testing and fire alarm purposes, becomes a major task, since the watchrooms must always be manned. With 37 alarm terminating watchrooms it means that the MFB must maintain some 160 capable firefighters just to watch the alarm termination panels.

* A glossary of terms is given in Appendix 8

4. THE NEW SYSTEM - GENERAL

To coincide with the opening of the MFB's new headquarters building, a major upgrading of the entire communications system is now underway. METCOM, (for METropolitan fire brigade COMmand and control system) is based on the use of dual Interdata 7/32 computers installed at Eastern Hill (Melbourne's central fire station and Brigade Headquarters). This is the hub of two interrelated systems: an alarm monitoring system embracing each of the 5250 subscribers; and a command and despatch system at the Control Centre.

5. THE ALARM MONITORING SYSTEM:

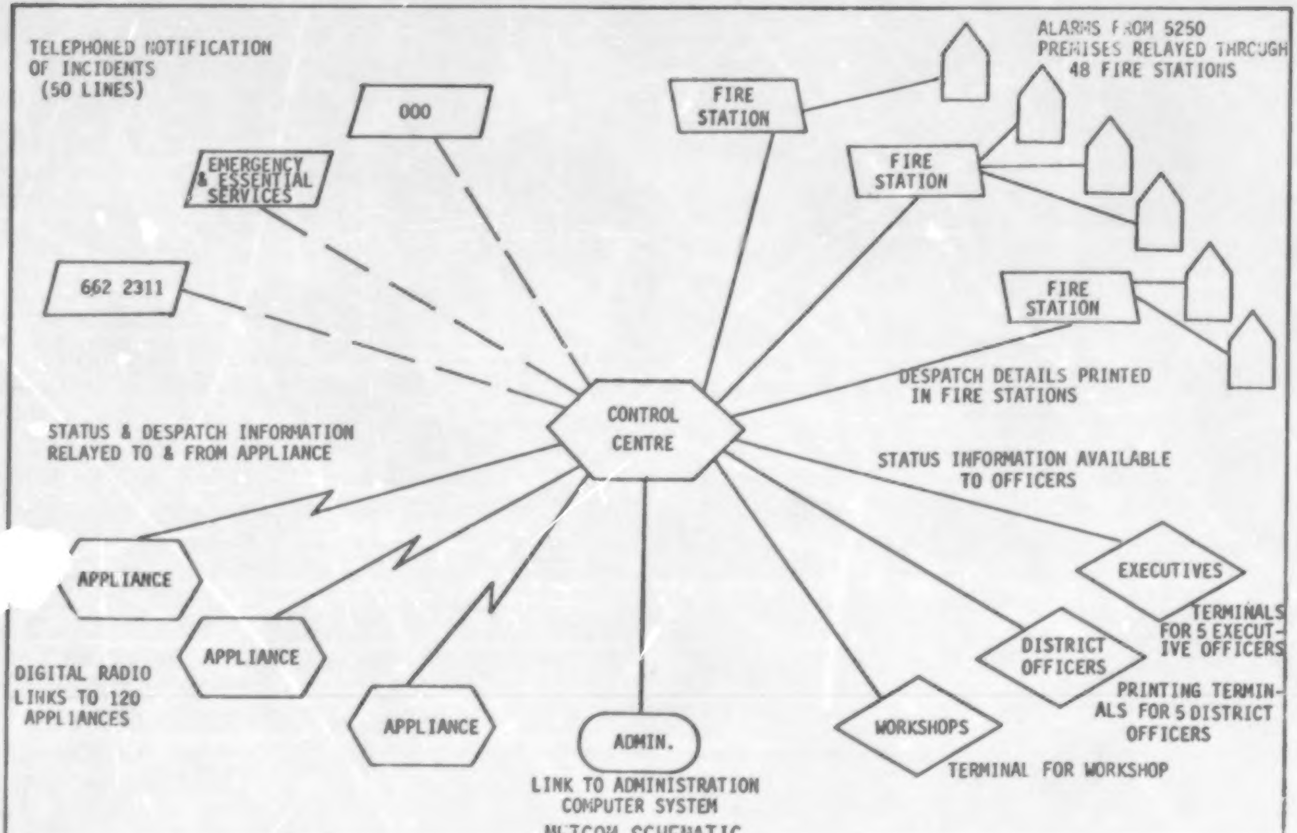
The telephone handset unit which is located in each fire alarm box or subscriber alarm panel is to be replaced by a small unit known as an "Alarm Interface Unit" (AIU). This will be connected by the existing leased telephone line to the nearest fire station. In each station will be installed a unit known as the Fire Station Equipment (FSE).

The Alarm Interface Unit is capable of sending and receiving up to seven bits of information. Up to eight units may be connected to the one pair of leased telephone lines. As currently being implemented the AIU will transmit to the FSE the following statuses:

a) Real Alarm
b) Test Mode
c) Alarm System Fault
d) Battery Fault

The remaining bits will be utilised later to monitor such items as the pressure in the pipes to sprinkler systems, and thus aid in the reduction in the very high rate of turnout due to false alarms from the automatic alarm systems.

In the present implementation only one bit is being transmitted from the FSE to the AIU. This will be transmitted when there has been a successful test. The method of testing is as follows: The person in charge of testing at the subscriber premises will turn a key to put the AIU into test status. He will then perform the test, and the FSE will return the bit indicating successful


TELEPHONED NOTIFICATION
OF INCIDENTS
(50 LINES)
000
EMERGENCY & ESSENTIAL SERVICES
662 2311
FIRE STATION
FIRE STATION
FIRE STATION
ALARMS FROM 5250 PREMISES RELAYED THROUGH 48 FIRE STATIONS
DESPATCH DETAILS PRINTED IN FIRE STATIONS
CONTROL CENTRE
STATUS & DESPATCH INFORMATION RELAYED TO & FROM APPLIANCE
STATUS INFORMATION AVAILABLE TO OFFICERS
APPLIANCE
APPLIANCE
APPLIANCE
DIGITAL RADIO LINKS TO 120 APPLIANCES
ADMIN.
LINK TO ADMINISTRATION COMPUTER SYSTEM
WORKSHOPS
TERMINAL FOR WORKSHOP
DISTRICT OFFICERS
PRINTING TERMINALS FOR 5 DISTRICT OFFICERS
EXECUTIVES
TERMINALS FOR 5 EXECUTIVE OFFICERS

METCOM SCHEMATIC

test; this will cause a LED indicator to flash on the unit, indicating a successful test. The flashing will be reset when the key in the unit is switched out of test status.

The FSE (fire station equipment) is actually a standalone unit, based on dual microprocessors. The functions of the unit are as follows:

a) To monitor each of the AIUs connected to it once every second. Each FSE may scan up to 1024 AIUs in one second although the maximum number in any fire station at the present time is 650.

b) To relay any change in status of any AIU to the Control Centre.

c) To initiate the "successful test" impulse to one AIU.

d) To receive messages from the Control Centre regarding the despatch of appliances: To each FSE is attached a small printer, and on this are printed the details of an incident for the appliances.

e) To act on instructions from the Control Centre to turn on the PA system for audio confirmation of despatch details, to turn on lights, open key cabinet, to sound the fire alert tones or bells within the station.

f) To monitor the status of battery supplies, power supplies, key cabinet and other sundry items, and relay changes of status to the Control Centre.

g) To recognise that it may no longer be connected to the Control Centre and to operate in complete standalone mode automatically reverting to terminal status when the connection has been re-established.

A schematic of the Fire Station Equipment is given in Appendix 1.

6. THE CONTROL CENTRE

The hub of the system is the Control Centre. Control Centre consists of two rooms, one which contains the computer and common radio, telephone, and radio equipment; and the other, the operations room.

6.1 DETAILS OF THE COMPUTER EQUIPMENT

The computer configuration consists of two (2) independent Interdata 7/32 systems, labelled the primary system and the standby system. Each system is able to handle all critical functions of the Alarm Monitoring and Command and Control process regardless of the state of the other system. To accomplish this task, the two systems are identical as far as critical processing components are concerned (see Appendix 2). The only difference between the systems is the addition of a line printer on the standby processor.

Each processor is equipped with the following items:

1) 256K bytes of 750ns core memory

2) Parity control

3) Power fail/auto restart

4) Universal clock module

5) Bootstrap loader

6) 64K bytes of multiport memory for computer-to-computer communications

7) A local VDU interface for 15 high speed asynchronous local lines (9 for the VDU terminal/keyboards and 6 for the status VDUs). The subsystem comprises of 15 separate interfaces each with double receive character and transmit character buffers. The method of operation is on a character by character basis operating at full duplex at a data rate of 9600 baud.

8) A local printer interface capable of handling two asynchronous lines at 300 baud for two local logging printers.

9) A communications subsystem used to communicate with:
   - . Up to 64 fire stations at 300 baud
   - . Five district stations and workshop at 300 baud

10) A console input/output device and controller. This device is the primary communication link between the programmers and operators and the operating system. Attached to the console devices is a paper tape reader. This provides a medium speed input device for loading diagnostic programs.

11) A special equipment interface subsystem. The special equipment interface provides interconnection between the computer and the following subsystem components:

    1) Time generator
    2) Multi point selection switch
    3) Special equipment - operation of 4 contact closures and detection of 4 key depressions

Two disc drives/controllers of 65MB capacity and 0.8MB/second transfer rate are connected onto switchable buses. In typical operation one disc is connected to each processor. All critical files are duplicated on the two drives and all transaction updates are to both drives via the shared memory.

The magnetic tape subsystem consists of a control unit and two tape drives switchable between processors. The subsystem is capable of both reading and writing tapes in industry compatible format. The main function of the tape subsystem is as a convenient method of data storage and system backup.

There is one (1) printer and controller on the standby processor. The printer is the impact type capable of producing multiple copies on standard printer forms at 300 lines per minute.

The standby processor continually monitors the primary processor. If a fault is detected then the standby machine will subsequently be automatically booted and take over the on-line function of the Command and Control System. The communications switch under control the the processors automatically switches over all the lines to the standby machine. Indication of a changeover is sent into the Alert Subsystem of the Application Programs. Reversion to the primary processor is manually performed. This device is being implemented by digital I/O channels between the processors capable of controlling the communications switch.

A Communication Switch provides the capability of switching all communication lines from one system to the other. The switch is under control of the active processor but manual override is provided. The inputs from all lines are channelled into both processors, regardless of their state, and only the output is switched between processors. This enables testing of the standby machine to be performed with live data independently of the on-line machine.

A total of nine VDU/keyboard terminals and six VDUs without keyboards serve as the primary medium for accessing the command and control system. The terminals are able to operate in both batch mode and character by character mode. The terminals consist of both rack mounted and desk top units as follows:

1) 12 rack mounted terminals - consisting of six working positions and six status terminals mounted in the consoles.
2) three desk top terminals in local offices.

Eight low speed hard copy printers, one with keyboard and the others without keyboards, serve as:

- Two local logging printers in the Control Centre
- Five printers, one in each of the District Offices
- One printer with keyboard at the Workshops

6.2 THE COMMAND AND DESPATCH SYSTEM

Using computers for fire brigade mobilisation is a relatively new sphere of activity, although their use for police and ambulance systems is more widespread. Numerous CAD (Computer-Aided Despatch) systems are now in operation with varying degrees of success, throughout the U.S. and to a much lesser extent, in the U.K.

The functions of a fire brigade CAD system may be stated as follows:

a) Keeping an up-to-date status of all current calls.

b) Keeping the up-to-date status of all appliances, i.e. in station, available on radio, at scene, out of commission etc.

c) Recording all details and status of all calls for historical and statistical purposes.

d) Enabling the rapid transmission of call details to fire stations and appliances.

e) Suggesting the assignment of the nearest and most appropriate appliances to any call.

f) Providing up-to-date status information to executive officers.

g) Enabling fire appliances to get to the scene of incidents rapidly and accurately.

6.3 THE OPERATOR'S CONSOLE

Six operator consoles are provided, one of these being for the supervisor. An operator, sitting at a console position has in front of him the following items:

To his left:	a VDU and Keyboard, used for the input of dynamic call and appliance information.
To his right:	Two VDUs One containing the Call Status; i.e. information concerning the status of all outstanding calls at this point in time; The other containing the Appliance Status; i.e. information indicating the disposition of all appliances at this point in time.
In front of him:	A radio panel enabling communication with three radio channels, i.e. his ability to speak to appliances in the field;

A telephone panel; which caters for 15 incoming public lines, six PABX lines, and 20 direct lines to essential services (i.e. to Electricity Commission, Board of Works, Airports etc).

A push-button panel controlling the set-up of direct PA communication to all fire-stations;

A short-term tape recorder with facility to instantly play back a previous conversation.

The operator's function is basically twofold:

a) To correctly identify the location of a call and to despatch the appropriate appliances to that call;

b) To communicate by radio with appliances in the field, and to enter information given by "wordback" into the computer system.

Communication between the operator and the computer is by means of commands entered on his operating VDU. The format of the operating VDU is given in Appendix 3 and a list of the commands is given in Appendix 4.

6.4 CALL AND APPLIANCE STATUS:

An example of the Call Status is given in Figure 2.

The Call Status contains a list of all active fire calls, and is able to indicate their status to an operator at a glance. Only calls to which appliances are still assigned are listed. Calls which are not yet under control are displayed in red; calls whose status has not yet been ascertained are displayed in yellow; calls which are under control are displayed in white. Colour conventions are likewise used to indicate the status of appliances assigned to the call: appliances on the way, on scene, or having left the call.

An example of the Appliance Status is given in Figure 3.

The Appliance Status indicates at a glance the status of all major appliances and all assigned appliances. Hose carriages, mobile tenders, ladders, pumps, snorkels, and salvage vans are regarded as major appliances and always appear on the Status whether assigned to a call or not. Less frequently used appliances such as the Breathing Apparatus van, the Lighting Unit, or the Hose Layer, only appear when assigned. Additional appliances, such as officers' cars, also only appear on the display when actually assigned to the call.

These two status screens have been designed in order to enable operators and executive officers to obtain immediate understanding of Brigade's disposition, and to be able to make rapid and accurate decisions in an emergency. For this reason VDUs are located in the offices of four executive officers so that this information is always and readily available to them.

The system basically consists, as has already been indicated, of two inter-related subsystems: an alarm-monitoring system and a command and despatch system. For the purposes of this discussion we shall consider them separately.

6.5 ALARM MONITORING SUBSYSTEM:

The 5250 alarm points around the metropolitan area are connected to micro-processors at the 48 fire stations. The micro-processors relay to the Control Centre computers any change in status at any one of these points. The Control Centre computers maintain details of all subscriber alarm points and of their status. Once a change in status is received by the control centre it will check the validity of the change it has received and then the nature of the change. It may be of four basic types:

a) A genuine alarm

b) A fault in the connected fire alarm circuit

c) A fault in the attached battery

d) A test

```
GF  /UC 10.43 1046 EXC LAVERTON DOUGHERTYS RD
                       H45 H47 H46
PW  /UC 12.04 1061 EXC NORTH MELBOURNE VICTORIA ST/ABBOTSFORD ST
                       H48A
    FA  12.16 1066 PFA CLAYTON MONASH UNIVERSITY
                       H25 H31
GF  /UC 12.20 1067 D24 MACLEOD DUNVEGAN CRES
                       H2 H3
    INV 12.26 1069 EXC BRIGHTON NORTH ROAD* BOAT ON FIRE
                       C35 H37
GF  /UC 12.28 1070 EXC ALBION BALLARAT RD* OVERPASS
                       H44 H43
CF  /UC 12.30 1071 EXC GLEN WAVERLEY VIEWMONT RD
                       H28 H29
    FA  12.31 1072 SPR PORT MELBOURNE HMAS LONSDALE
                       H38 H39
BF  /AR 20.04 1106 EXC BRAYBROOK CRANWELL ST/BUTLER ST* FACTORY FIRE
                       M43 P47 STH47 H47 H41 BA HL P1A SN45 P42 SN1
                       CFO DCO SV3 LU MC H10A H4 RT1
```

<u>Figure 2</u> - <u>CALL STATUS</u>

(See Appendix 6 for explanation)

```
H1A          H9           H4      H18     H24     H41            CFO
H1B                       P4                      H42A           DCO
H1C  XH47    H10A                 H19     H25     H42B
C1           H10B         H5                      P42            AC1
P1A          M10          M5      H20     H29                    AC2
P1B          SN10                                 H43    1070
M1                        H6      H21     H31     M43    1046    DS1
SN1                                       M31                    DS2
HL1                       H11     H22             H44    1046    DS3
T1                                        H32
LU1          H35          H12     H23             SN45           DO1
BA1          C35                  SV23    H33                    DO4
RT1          T35          H13             M33     H46    1046    DO47
MC1                       M13     M26                            DO23
             H38   1072                   H34     H47   *1046    DO25
H2                        H15     H27     P34     P47
C2           H39   1072           P27
T2           P39          H16             H36     H50    1068
                                  H28     H37
H3   1067    H48A                                 H51            XH47
SV3          H48B                 H30             M51
T3                                M30
                                                  H52
```

Figure 3 APPLIANCE STATUS

(See Appendix 7 for explanation)

These differentiations are important in the assigning of appliances to the call. If there is a fault in the battery then it is only necessary to notify the appropriate maintenance company; there is still sufficient power in the battery to enable the alarm relays to operate. However, to a fault in the connected fire alarm circuit, one appliance will be despatched. It is probable that the cause is only a fault. It could however be something more serious, and in any case the property is unprotected until the situation is corrected.

In the case of a genuine fire alarm, then the full complement of appliances is despatched.

In the case of a test, the fact of the test is recorded.

Details of all alarms, false alarms and types of false alarms, and tests are kept within the computer in order to monitor trends and highlight defective alarm systems, and delinquent testers. Since the rate of increase in false alarms from automatic systems continues to rise, controlling of this trend becomes of great importance.

6.6 COMMAND AND DESPATCH SUBSYSTEM:

An operator will be alerted to an alarm in a subscriber premise by the word "ALARM" flashing in the bottom left hand corner of his screen; if the operator position has been inactive then the keyboard buzzer will also sound. The operator gains control of the alarm by depressing an "Acknowledge" function key on his keyboard. Details of the location of the alarm and the appliances to be assigned appear on the Operating VDU.

The operator ratifies the computer's appliance assignment, and then by depressing the special "Despatch" key actual despatch operations will commence:

a) The computer will cause the connection of direct voice lines from the console to each of the stations from which appliances are to be despatched; two alert tones (a modern and more pleasant version of the traditional strident bells) will be sounded for 7 seconds within the stations, and the operator will then announce the despatch details over the stations' PA systems. Simultaneously, details of the call are transmitted in digital form via a separate data line to the fire station micro-processors for printing on the attached printer.

b) Each fire station will acknowledge receipt of the alarm details by depressing its "Station Acknowledge" key.

Whilst all this has been happening the call has appeared on the Call Status Display. As appliances have been assigned both Call and Appliance Status Displays have been updated indicating the disposition of the appliances from "preliminary assignment", through "assigned", to "on the way".

Calls received by other than the Alarm Monitoring System:

Alarms may also be received by telephone from the public, on direct lines from Police, airport, etc., or as "passerby" alarms whereby someone actually comes into fire station to give the call details.

In such cases, when the operator has received the call details, he types in details to the Operating VDU. For example:

```
IN  EXC  109 GR TOOR/BR *HF,
```

which may be interpreted as an exchange call to a housefire (HF) at 109 Grange Road Toorak near Bruce Street. (IN is the "INITIATE CALL" mnemonic.)

A Geographic database file is kept containing details of all suburbs (and their immediately surrounging suburbs), and the streets, intersections and landmarks within them. In our above example the computer will search for an intersection GRBR in Toorak, which it will then identify as being the desired location. This will then be displayed on the screen, the operator will verify that this is indeed the desired location and then depress his "Assign" function key.

Should there be more than one intersection within the area which fulfills the criteria, then the alternatives will be listed. The appropriate one is selected by number and the "Assign" function proceeds.

Should the desired intersection not appear in the nominated suburb or if the presented solution is not the correct one, by depressing the "Auxiliary" function, the computer will display possible solutions in surrounding suburbs. Should this procedure still yield no acceptable solution the facility exists to interrogate the street directory, in much the same way as one would use a street index at the back of the normal street directory you carry in your car.

6.7 UPDATING THE PROGRESS OF A CALL:

An appliance, once it reaches the call-scene, will radio back a message:

"Hose Carriage 1A on scene"

The operator enters this to his screen as

OS H1A

Entry of this command will cause details of the call, if they are not already there, to be displayed on his screen.

The most important update however is via the wordback. Typical wordbacks are of the form:

From Hose Carriage 1A; Station Officer Smith,
Grassfire under control

From Pump 9; District Officer Jones; House fire
not yet under control

From Mobile Tender 4; Station Officer Black,
False Alarm

From Turntable Ladder 7; Station Officer Gray,
Factory fire, Well Alight, Assistance required

These items are typed into the computer as follows:

```
WB  H1A,SO Smith, GF, UC
WB  P9,DO Jones, HF, NY
WB  M4,SO Black, FA
WB  T7,SO Grey, BF, AR,*WELL ALIGHT
```

Details of all wordbacks are incorporated into the Call History. At any time it is possible to obtain a complete chronological history of the progress of a call, wordbacks and action taken. In case of later enquiry or possible legal complications, the details of the call history can be matched with details of the audit log, and the longterm tape recorder which records all voice communication. Time is synchronised by an independent clock from which the computer takes its time (it is not fed in by an operator), and which emits data and time to the wall clock and the tape recorder. As a final check the Australian Telecom spoken time facility is also recorded on the tape recorder.

When appliances have completed their attendance at a call they radio back: e.g. "Hose Carriage 27 and 4 men returning to station". This is entered to the computer as

```
RS H27,4
```

Some of the other commands used in the system work as follows:

c(AS) Assign Appliances

This enables the assignment of specific appliances or types of appliances to a call, as opposed to the normal automatic assignment of appliances: e.g. `AS SN, C1`
Assign the nearest available snorkel, and the combination ladder from station No. 1.

c(UA) Unassign Appliance

The despatch operator, or executive officer may disapprove of the computer's assignment of an appliance to a call, in which case this command gives him the facility to unassign the appliance.

c(IS) In Station

Appliances returning from a call or drill, must notify their status as "In Station".

c(ST) Send an appliance as a Standby to a given station

e.g. ST H27, FS47
The normal hose carriage from fire station 47 is attending a fire, and other appliances in the vicinity are engaged. In such a circumstance the hose carriage from station 27 is sent as a standby hose carriage to station 47. It takes a new designation as Standby Hose Carriage 47 (XH47).

c(SR) Standby Return

An appliance which was on standby at another station, is now returning to its own station.

c(CO)

This allows a comment or any narration to be entered into the call history.

c(DC) Display Call

This enables details of any call, current or within the past month, to be displayed on the operators VDU. A sample of a DC is given in Figure 4.

c(NA) Notify Authorities

At large fires it may be necessary to inform various local statutory or emergency bodies. This command enables the recording of the fact that such bodies have been notified.
e.g. NA MMSW, SEC, D24, CFO
The Board of Works, the State Electricity Commission, the Police and the Chief Fire Officer have all been informed.

c(DR) Drill

Appliances which are out on Drill must notify their status which is then indicated on the Appliance Status. Appliances at Drill are usually available for assignment to calls, although they are not available in station. Under such circumstances call details are given via the radio. In a later phase when digital radio is implemented, call details will be actually printed within the appliance from the printing status head.

c(TC) Appliances are occasionally sent to the Training College for instruction of trainees

In general such appliances are not available for primary assignment, but are available for assignment if required. It is thus important that the correct disposition of appliances is known at all times.

c(IC) In-Commission

An appliance is available for assignment.

c(OC) Appliance is Out-of-Commission

Such an appliance is not available for assignments.
e.g. OC P52 Pump 52 is out-of-commission.

c(DV) Designate Vehicle

There is a difference between a vehicle and an appliance. "Appliance" is the title given for the time being to a vehicle e.g. H1A = Station No 1's first hose carriage. In fact the MFB has about fifteen spare hose carriages, which are called into service when the normal hose carriage is for some reason out of commission. A spare hose carriage (whose vehicle or asset number is its basic designation) will assume the appliance number. e.g. To remove an appliance from a station to the workshops and replace it with a spare:

DV H16, V151
DV V180, H16
MV V151, FS16, WS

which means:

V151 is now no longer known as H16
V180 is now given the designation H16
V151 (the old H16) is moved from station 16 to the Workshops.

c(MV) Move Vehicle
c(VS) Vehicle State

A constant problem faced by the Brigade is that of knowing the location of its vehicles. Especially on a Monday morning the workshops may find that there has been quite a reshuffle of vehicles during the weekend. The c(MV) command enables the movements of all vehicles to be recorded, the c(VS) command enables executive officers or the Mechanical Manager to know at any time the location of vehicles.

c(IT) Isolate Alarm Termination
c(DT) Deisolate Alarm Termination

c(IT) gives the control centre the ability to "isolate an alarm termination", i.e. make it inactive and ignore any alarms received from it". c(DT) reserves the function.

c(AA) Action Alert

This enables an operator to obtain the particulars of an internally generated alert e.g. an appliance has been assigned to a call, but no "station acknowledgement" has been received; or the data line to a fire station has become inoperative; or a subscriber alarm has been in test status for an inordinately long term.

7. APPLIANCE ASSIGNMENT:

Apart from the fact that the entire system must physically run 24 hours a day 365 days a year and be therefore of the utmost reliability, the most crucial area of operation must be in appliance assignment.

In many overseas brigades appliance assignment becomes a formidable problem, as every possible appliance configuration is manually calculated for each city location. All this information must be kept in machine form for appliance assignment, but also in hardcopy form for manual backup in case of machine failure. In a city the size of Melbourne such a system would be utterly impracticable, since the file maintenance problems would be forbidding. Fortunately the MFB has kept its rules simple. The previous method of defining assignment rules was to take a

Melbourne Street Directory, and felt-pen in on it the various assignment areas, and within these areas the two stations considered nearest to them. Thus when a call was received, and its location found on the street map, the two nearest stations were already known. The complications arise in what to do when those nearest two hose carriages are not available. Again the rule is basically a simple one: It is that the nearest two available appliances are sent. The city has been divided in 250 assignment areas, which indicate the two, three or more nearest stations to all places within that assignment area. Should appliances not be available at those listed stations, then the nearest appliance is calculated using a formula based on the known distances of all stations from one another, and the known nearest stations to the call location.

The ability of the computer to provide the exact map-reference of a call is an important feature. This information is printed on the despatch details, and is always associated with the call details. Thus it ensures that appliances all go to the same location (not different ones as in the past), and enables the computer to readily identify possible duplicate calls.

8. CAPTURING OF THE DATA:

There are some 50,000 streets and 120,000 intersections in the MFB area. The accurate capture of these could have presented a problem. This was solved in a novel fashion using a digitiser technique.

The street index from the back of the street directory was captured. Each map then was placed on reference plate. The map and grid references were identified. Assignment areas and suburbs were likewise delineated. Then the name of each street was entered into the data capture system, and the intersections on it marked with the digitiser. Thus tracing the routes and intersections for each street enabled a double verification function to be achieved:

a) Where the streets met must be the intersection

b) Each of the streets, as entered for the digitising could be verified against the street index data which had been entered earlier.

9. SYSTEM BACKUP:

No matter how sophisticated our hardware or software, only one thing is certain, and that is that it will occasionally fail, To avoid this there are dual power supplies, dual computers, dual discs, redundant consoles, dual lines to stations etc.; each fire station is capable of standing alone. However in spite of the mechanical backup, occasionally it will be necessary to revert to manual means. A console printer lists the disposition of call and appliance status after every ten appliance movements, so that a manual system may be rapidly established. The district station will ensure that men are sent to each station to monitor the standalone microcomputers and relay any alarms. Equally the facilities exist for events which occurred during machine failure to be re-submitted to the system.

10. FUTURE ENHANCEMENTS:

10.1 DIGITAL RADIO:

Once the control system is operating it is planned to place digital radio transceivers in each of the appliances. These microprocessors will enable appliances to receive, actually printed within the appliance, details of calls, appliance assignments and re-directions.

10.2 HAZARDOUS CHEMICALS:

The problem of what to do with an unknown chemical is one which often confronts firemen at the fire scene. The ability to be able to interrogate a National Chemicals Database from the Melbourne Control Centre is one which greatly appeals to firemen.

10.3 STATISTICAL AND ADMINISTRATIVE APPLIACATIONS:

This work is to be achieved quite separately from the operational CAD system using an ICL 2903. It makes great sense that the two systems, the operational and the administrative, should talk to and complement each other. Fire report details will be relayed from the operational machine to the administrative machine for statistical analysis. Similarly information on men available can be utilised by the operations machine for assignment of men to special duties.

11. CONCLUSION:

The Metropolitan Fire Brigades Board of Melbourne has embarked on an ambitious program to upgrade its facilities and services. Although the economic benefits are considerable, perhaps more important will be benefits gained in more accurate control of subscriber alarms, warnings of faulty equipment, and ability to despatch appliances more rapidly and accurately than has been possible in the past.

The system is being implemented by:

The Metropolitan Fire Brigades Board of Melbourne

E. Johnson	President
N. van Every	Chief Fire Officer

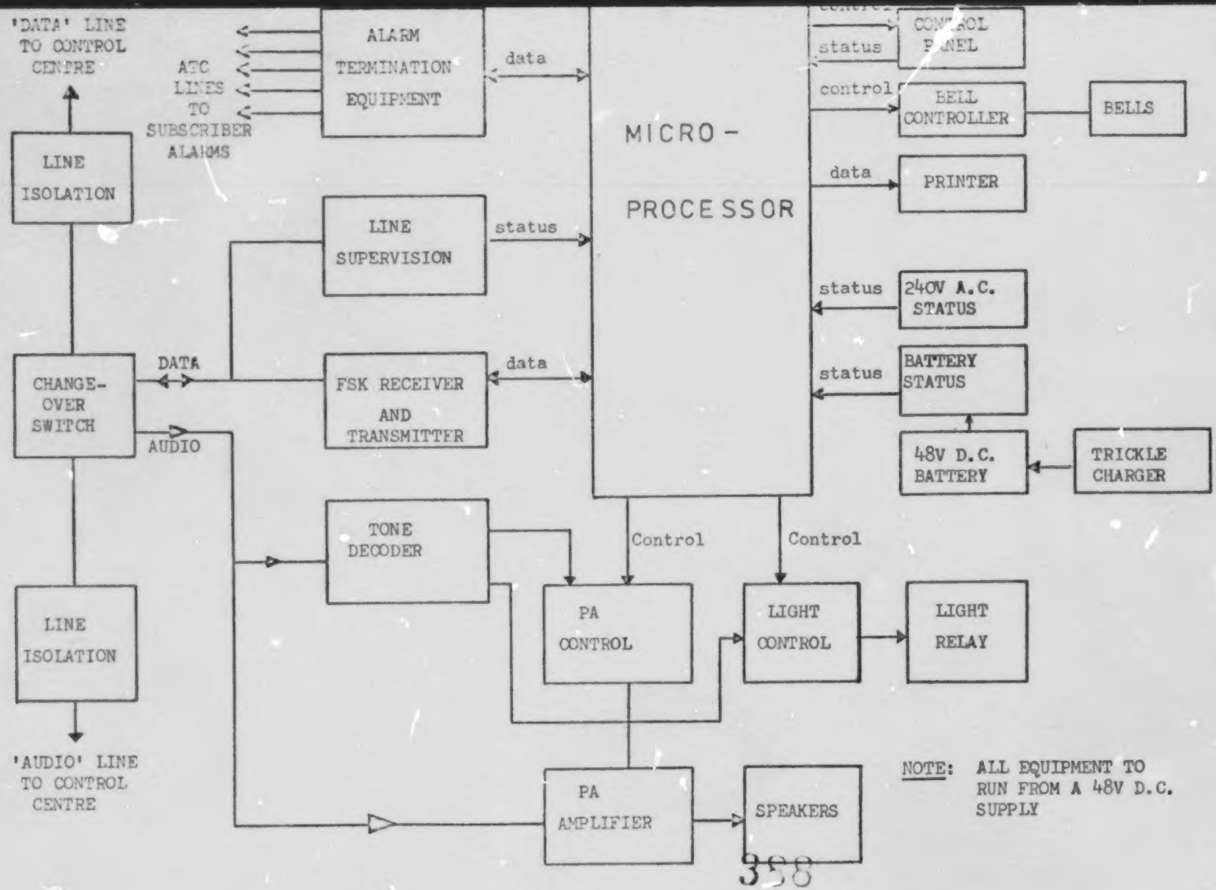
Consultants: Public Management Services Ltd

Contractors: Philips Telecommunications Manufacturing Company Ltd, as prime contractor, in association with:

EDL (Australia) Ltd	(telemetry)
Systems and Programs (NZ) Ltd	(computer software and hardware)
Perkins Elmer Data Systems Limited	(Interdata computers)

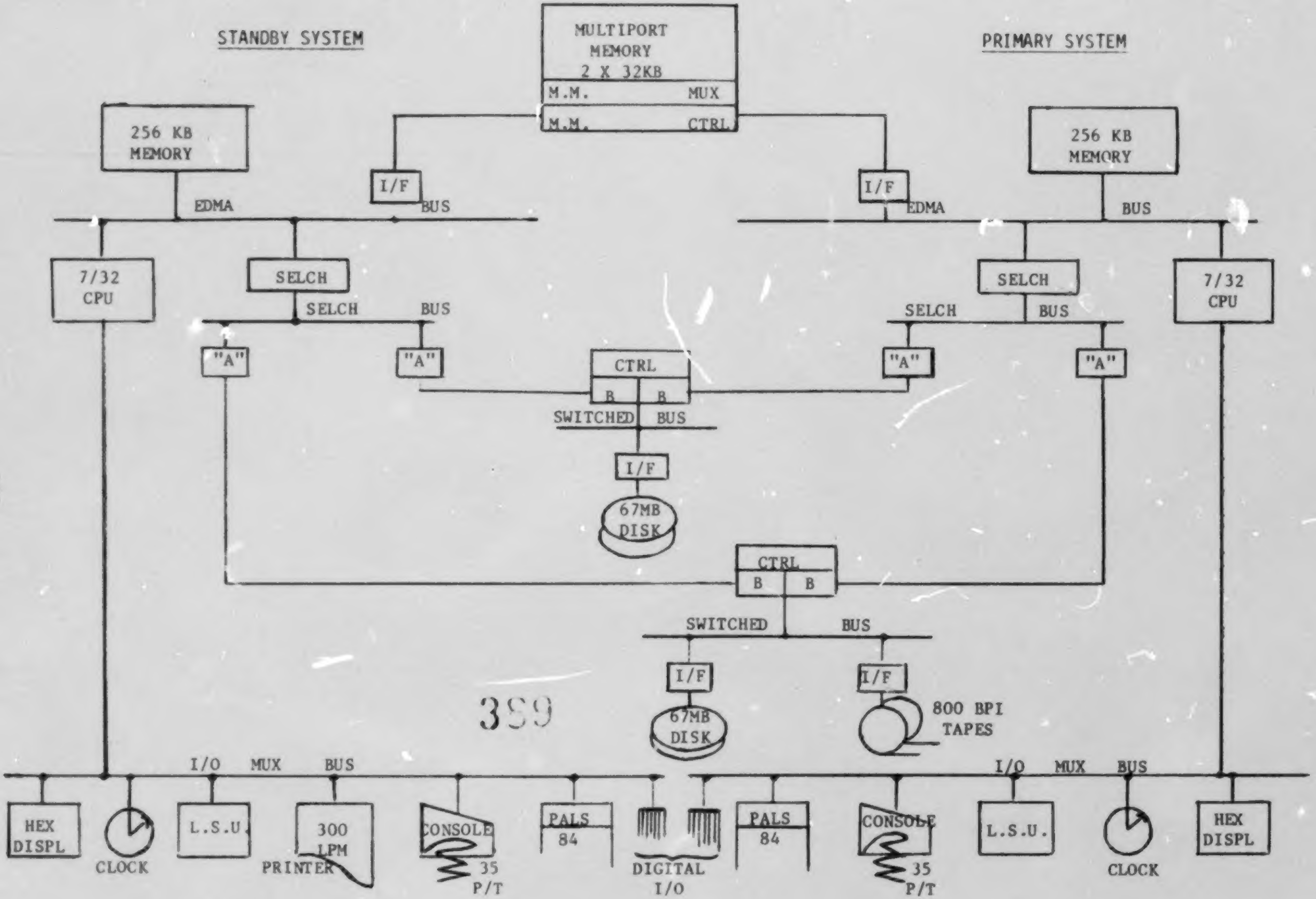
APPENDICES

1. Schematic of Fire Station Equipment
2. Schematic of Computer Equipment
3. Format of Operator's Operating VDU
4. List of Commands
5. Detailed Call Display
6. Call Status
7. Appliance Status
8. Glossary


'DATA' LINE TO CONTROL CENTRE
LINE ISOLATION
ATC LINES TO SUBSCRIBER ALARMS
ALARM TERMINATION EQUIPMENT
data
MICRO-PROCESSOR
control
CONTROL PANEL
status
control
BELL CONTROLLER
BELLS
data
PRINTER
LINE SUPERVISION
status
status
240V A.C. STATUS
CHANGE-OVER SWITCH
DATA
FSK RECEIVER AND TRANSMITTER
data
status
BATTERY STATUS
AUDIO
48V D.C. BATTERY
TRICKLE CHARGER
TONE DECODER
Control
Control
PA CONTROL
LIGHT CONTROL
LIGHT RELAY
LINE ISOLATION
'AUDIO' LINE TO CONTROL CENTRE
PA AMPLIFIER
SPEAKERS

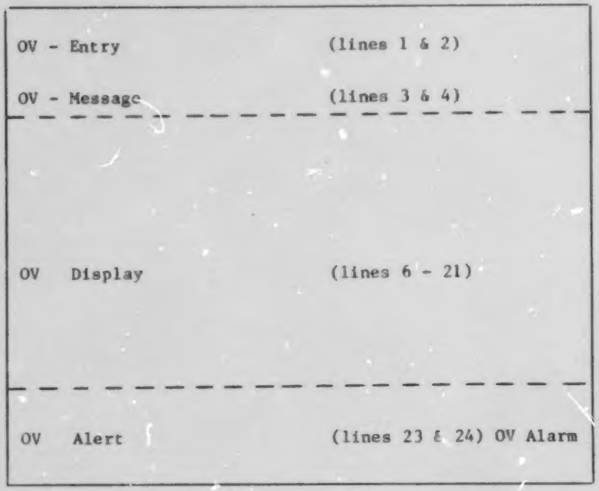
NOTE: ALL EQUIPMENT TO RUN FROM A 48V D.C. SUPPLY

APPENDIX 1 FIRE STATION EQUIPMENT SCHEMATIC


STANDBY SYSTEM
MULTIPORT MEMORY 2 X 32KB
M.M.
MUX
M.M.
CTRL
PRIMARY SYSTEM
256 KB MEMORY
I/F
EDMA
BUS
I/F
EDMA
256 KB MEMORY
BUS
7/32 CPU
SELCH
SELCH
BUS
SELCH
SELCH
BUS
7/32 CPU
"A"
"A"
CTRL
B
B
"A"
"A"
SWITCHED
BUS
I/F
67MB DISK
CTRL
B
B
SWITCHED
BUS
I/F
I/F
67MB DISK
800 BPI TAPES
I/O
MUX
BUS
I/O
MUX
BUS
HEX DISPL
CLOCK
L.S.U.
300 LPM PRINTER
CONSOLE
35 P/T
PALS 84
DIGITAL I/O
PALS 84
CONSOLE
35 P/T
L.S.U.
CLOCK
HEX DISPL

APPENDIX 2 SCHEMATIC OF COMPUTER EQUIPMENT

APPENDIX 3: FORMAT OF OPERATOR'S OPERATING VDU


OV - Entry
(lines 1 & 2)
OV - Message
(lines 3 & 4)
OV Display
(lines 6 - 21)
OV Alert
(lines 23 & 24) OV Alarm

Detail of OV (Alert) and OV (Alarm):

line 23	1.Alert 1	2.Alert 2	3.Alert	**ALARM*
line 24	4.Alert 4	5.Alert 5	6.Alert	
	1 24	25 48	49 72	73 80

lines 1 and 2 referred to as OV (Entry)
lines 3 and 4 referred to as OV (Message)
line 5 a separator line of hyphens
line 6 - 21 referred to as OV (Display)
line 22 a separator line
lines 23 and 24 referred to as OV (Alert) with a subsection for OV (alarm)

- OV (Entry) is used for entry of commands.
- OV (Message) is used for the display of messages or errors in the submitted command.
- OV (Display) is used for the display of data associated with the command, e.g. a call, street directory etc.
- OV (Alert) is used by the Alert Subsystem to display alert information.
- OV (Alarm) is used to flash a 'ALARM' message when one is received from the fire station equipment.

In principle, data can only be entered into OV (Entry). There are only two exceptions: f(DS) where data may be entered into selected fields on the OV (Display), and f(FM) where the full screen format is used.

APPENDIX 4: LIST OF COMMANDS

There are four types of command as detailed below.

1. Console Commands:

These commands are entered from the VDU keyboards.

c(AA)	Action Alert
c(AS)	Assign Appliances
c(CO)	Comment
c(DC)	Display Call
c(DT)	De-isolate Termination
c(DR)	Drill
c(DV)	Designate Vehicle
c(EC)	End Call
c(IC)	In Commission
c(IN)	Initiate Call (Non-automatic)
c(IS)	In Station
c(IT)	Isolate Alarm Termination
c(LC)	List Call History
c(MC)	Modify Call
c(MV)	Move Vehicle
c(NA)	Notify Authorities
c(OC)	Out of Commission
c(OK)	Open Key Cabinet
c(OS)	On Scene
c(RS)	Return to Station
c(SR)	Standby Return
c(ST)	Standby
c(TC)	Training College
c(TO)	Turn Out
c(UA)	Unassign Appliance
c(UC)	Unable to Complete
c(VS)	Vehicle Status
c(WB)	Wordback

2. Functions:

These commands are entered by depression of a function key on the VDU keyboards.

f(ACK)	Acknowledge Alarm
f(AP)	Appliance Status
f(AS1)	Assign 1st level
f(AS2)	Assign 2nd level
f(AUX)	Auxiliary function
f(CS)	Call Status
f(DS)	Display Street Directory
f(FM)	File Maintenance
f(IN)	Initiate Non Automatic
f(MAP)	Display Map
f(PB)	Page Backward
f(PF)	Page Forward
f(PR)	Print Display Contents of VDU

3. Fire Station Equipment Commands:

These are messages received from a fire station.

s(AC)	Alarm Change of Status
s(CT)	Configuration Table
s(KC)	Key Cabinet Alarm
s(OF)	Equipment "off the air"
s(ON)	Equipment "on the air"
s(SA)	Station Acknowledge

4. Input Register Commands:

These are commands received by depression of push buttons on the consoles.

i(DE)	Despatch Appliances
i(CL)	Clear Crossbar

```
1106  20.04  EXC   BF BRAYBROOK CRANWELL ST/BUTLER ST    MEL 27C10 STATUS F/UC
 AS   20.04        M43 P47
 WB   20.16  P47   SO HARBROW, BF, AR, *FACTORY 100 x 50 BA IN OPERATION
 NA   20.16        CSUP
 AS   20.16        XH47  H47  H41  BA1  HL1  P1A  SN45
 WB   20.32  H47   DO TROTTER,*FIRE SINGLE STOREY FACTORY 100 x 150 SOUTHERN
                              VEGETABLE OIL WELL ALIGHT
 NA   20.32        CFO  DCO
 AS   20.32        P42  SN1  CFO  DCO  SV3
 NA   20.40        SEC D24
 WB   20.51  P1A   SUPT WILSON* & SINGLE STOREY BUILDING 40m x 70m COMPLETELY
                               INVOLVED SNORKEL REQUIRED
 NA   20.54        MMBW
 WB   21.13  CFO   FIRE CONTAINED TO SINGLE BUILDING, LU REQUIRED
 AS   22.17        MC
 WB   22.19  CFO   SUPPLIES OF LIGHTWATER TO BE SENT ON
 AS   22.19        TR1
 WB   22.33  CFO   H10A WITH FULL CREW AND H4 WITHOUT DO
 AS   22.38        H10A H4
 WB   22.56  CFO   U/C
 WB   23.08  CFO   DO AND 7 MEN FOR FIRE DUTY
```

APPENDIX 5: DETAILED CALL DISPLAY

EXPLANATION TO DETAILED CALL DISPLAY

This display is called up on the OV (Display) portion of an operator's VDU, whenever he uses the commands DC, WB, OS, RS, TO.

Line 1: Call No. 1106 received at 20.04; an exchange call to a building fire (BF) in Braybrook, in Cranwell Street near Butler Street. Melway street directory reference Map 27, Grid C10. Current status is Fire Under Control.

Line 2: At 20.04 M43 and P47 were assigned to the call. (It is unusual that a mobile tender and a pump would be primary appliances to a call. However on this occasion the hose carriages were attending another fire).

Line 3: At 20.16 a Wordback from Station Officer Harbrow; "Building Fire, Assistance Required, Factory size 100m x 50m. Breathing apparatus in operation".

Line 4: At 20.16 the Control Superintendent is notified. (NA = Notify Authority).

Line 5: At 20.16 - a string of appliances is assigned to the fire. The district officer also goes to the fire.

Line 6: The District Officer gives his wordback the building is "well alight".

Line 7: The Chief Fire Officer and his Deputy are notified.

Line 8: Additional appliances, the Chief and Deputy are assigned to the fire.

Line 9: State Electricity Commission and Police are notified.

Line 10: The Control Superintendent, now the senior officerat the fire, gives his wordback, and asks for the Snorkel.

Line 11: Board of Works notified.

Line 12: Chief Fire Officer now in attendance and command gives his wordback.

Line 13: Mobile Canteen assigned to fire.

Line 19: Fire is under control.

APPENDIX 6: CALL STATUS - EXPLANATION TO FIGURE 2

Item 1: Call No. 1046, received at 10.43 by exchange telephone to grass fire (GF) at Doughertys Road Laverton. Hose carriages from Stations 45, 47 and 46 are in attendance and the fire is under control (UC).

Item 2: Call No. 1061 to a Petrol Washdown.

Item 3: Call No. 1066 to Private Fire Alarm at Monash University. It was a false alarm

Item 9: Call No 1072. False alarm on a sprinkler system.

Calls 1046, 1061, 1066, 1067, 1070, 1071 and 1072 would all appear in white.

Call 1069 would appear in yellow.

Call 1106, would appear in red with "BF/AR" flashing.

APPENDIX 7: APPLIANCE STATUS - EXPLANATION TO FIGURE 3

H = Hose Carriage
C = Combination Ladder
P = Pump
M = Mobile Tender
SN = Snorkel
HL = Hose Layer
T = Turntable Ladder
LU = Lighting Unit
BA = Breathing Apparatus Van
RT = Relief Tender
MC = Mobile Canteen
SV = Salvage Van
CFO = Chief Fire Officer
DCO = Deputy Chief Officer
AC = Assistant Chief
DS = District Superintendent
DO = District Officer

H1C: Hose Carriage 1C is standing by at station 47 as Standby hose carriage 47 (XH47)
H3: is attending call 1067
H38 & H39: are attending call 1072
H43: is attending call 1070
M43, H44, H46, H47 are attending call 1046
H47 however is covered by a standby - indicated by asterisk. (XH47 at bottom right hand corner)
H50 is attending call 1068

APPENDIX 8: GLOSSARY

1. Alarm: a) The wall box, sprinkler system, thermal system, push-button mechanism which is used for indicating presence of a fire.

   b) The receipt of information concerning a suspected fire.

2. Appliance: An appliance is the title given for the time being to a vehicle. e.g. H1A:Station No.1's first hose carriage. From time to time different vehicles may assume the appliance code.

3. Call: A "call" is a demand made on the fire brigade's services. Calls may be of two types: a "fire" or an "incident" (i.e. a non-fire).

4. Incident: An "incident" as used in Melbourne is a non-fire. e.g. a motor accident, a gas leak, a rescue, a petrol washdown.

5. Standby: When an appliance is away from its home station attending a call, an appliance from another station may be sent as "standby" to that station.

6. Subscriber: Organisations which have alarm systems connected directly by leased line to a nearby fire station, are known as "subscribers". Thus the term "subscriber alarm".

7. Termination: Subscriber alarms terminate at a fire station. Thus Eastern Hill fire station has 650 terminations. In most cases "termination" may be taken to mean "subscriber alarm".

8. Vehicle: A 'Vehicle' refers to the basic identification, or a static asset number of a vehicle. The vehicle number does not change. However various vehicles may from time to time assume different appliance numbers.

MANAGEMENT CONSULTANT - AID TO EFFECTIVE COMMUNICATION OR EXPENSIVE LUXURY?

AUTHOR: NEIL N McDOUGALL, B.A. B.C.A. *
Data Processing Section,
Wellington Polytechnic

ABSTRACT:

This paper examines the problem of communication and proposes that a management consultant can be an aid to effective communication. An outline of both the traps and benfits by employing such a consultant are thus detailed.

KEY WORDS AND PHRASES:

effective communication, management consultant, professional approach, management consultant for data processing.

C.R. CATEGORIES:

2.10, 2.11, 2.40, 3.50.

---

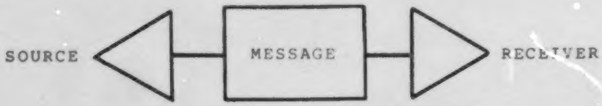
* Formerly of Clarke Menzies & Co., Chartered Accountants, Wellington.

MANAGEMENT CONSULTANT - AID TO EFFECTIVE COMMUNICATION OR EXPENSIVE LUXURY?

Neil N. McDougall BA, BCA

What is 'effective' communication?

All communication involves three elements:



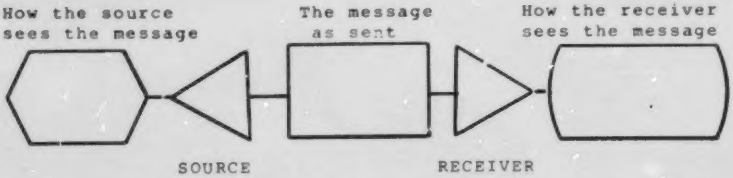
- We may have more than one receiver of the message as in a lecture hall, conference or classroom.
- We may have more than one message, but one must follow another if they are not to clash.
- There may be more than one source, as there is at this conference, but again to avoid confusion the sources must send their messages one after another.
- The message must contain some information.

From the receiver's point of view he may perceive the message clearly or distorted by 'noise'. We have looked at two causes of noise:

- multiple sources sending at the same time
- a mixture of messages, as when both the radio and TV are going at the same time.

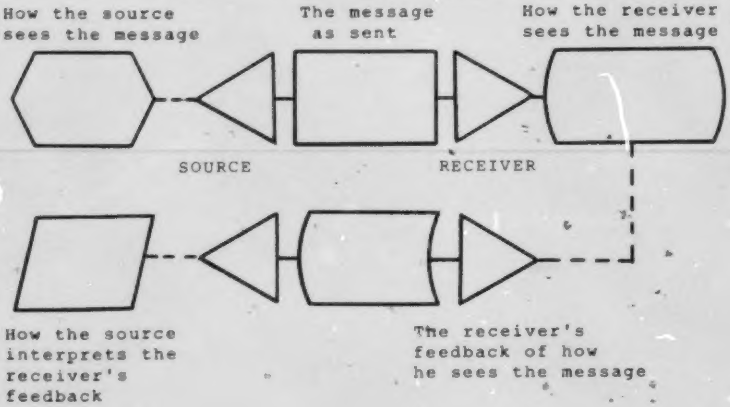
Then there is external noise to contend with. For example trucks passing, typewriters clattering, a beautiful secretary walking past the door; all of which diminish the clarity of reception.

Lets expand our model eliminating all noise interference


How the source sees the message
The message as sent
How the receiver sees the message
SOURCE
RECEIVER

We still do not have 'effective' communication, for both source and receiver see the message in a different light.

What we need is a two way model allowing feed back and modification of the images of the messages as seen by source and receiver, and the message itself.


How the source sees the message
The message as sent
How the receiver sees the message
SOURCE
RECEIVER
How the source interprets the receiver's feedback
The receiver's feedback of how he sees the message

Obviously communication is a complex process. Even without the threat of interference by noise it may take several iterations of the message - feedback process before source and receiver are seeing eye to eye. In political communication the messages are always slanted to project the meaning most advantageous to the party. But in the world of computers a lot of money can be wasted if effective communication does not exist.

The Requirements for Effective Communication

When to Communicate?

In Western Society people are inundated daily with incoming communication. At work it takes many forms: correspondence, trade literature, secretaries, managers, employees and clients to deal with. At home there are newspapers, radio, T.V., advertising material received through the mail, accounts, and most satisfyingly one's family pushing information at us. Most of this communication is probably ineffective.

Effective communication calls for an evaluation of the situation on which communication is needed.

The objective of effective communication is to cause behaviour that will produce the result sought by the communicator.

What to Communicate?

Tell a person everything and he will probably understand next to nothing about what you are trying to say. Select the factors that are important - that is what should be communicated.

How to Communicate?

- Having selected the information to communicate, it needs to be marshalled into a logical order, with facts supporting arguments.
- Next one must decide on the media in which to communicate these ideas. Should it be verbal, or written?
- At what level is the communication to be pitched? Is it for a General Manager or Goods Inwards Clerk? How much detail is required?
- How is the material to be presented?

How do you know you have Communicated?

Firstly feedback is vital. Without it you cannot tell whether all the important parts of your message have been received and understood. It allows for clarification of the meaning of the communication.

Secondly, when some action results. If you have communicated effectively it should be an action that is compatible with your evaluation of the original situation leading to the communication.

What is a Management Consultant?

What are the characteristics one expects to find in a good management consultant? Here is a brief list of attributes which should be present to some degree:-

- A manager, used to allocating and managing scarce resources
- A problem solver
- An experienced communicator at all levels
- A possessor of sufficient technical knowledge to understand the implications of choice relating to computer hardware and software
- A missionary in often unresponsive and even hostile territory
- A person experienced at introducing change
- A creative innovator
- A possessor of the drive to finish activities
- A person who stays calm and is impartial
- A possessor of wide range of experience in a variety of business situations
- Product oriented
- An optimist - he believes anything worthwhile can be done
- A pessimist - he never expects anyone to achieve their goals unless they have proven they can.

Where do Management Consultants exist?

In New Zealand management consulting organisations fall within three categories:-

(1) The individual working largely on his own but who has contacts for distributing workload on large assignments.

(2) Firms largely involved in the recruitment and placement of managerial personnel.

(3) The Management Services arm of large public accounting firms whose major role is in supporting the audit function of these firms.

One could include software houses and computer bureaus, but these organisations, while providing computer services, do not generally get involved initially as an aid to solving a firm's managerial problems.

Who hires Management Consultants in New Zealand?

The range here is surprisingly broad. As one would expect in a country where the government is heavily involved in economic activities such as communications, transport, banking and production as well as the provision of Social Services, government itself is a major user of consulting services. The use of management consultancy has occurred at many levels within the government machinery.

- Government departments e.g. Post Office, Department of Health.
- Local bodies such as hospital boards and borough councils.
- Government owned commercial entities.

Private enterprise firms also use Management Consultants. While larger firms sometimes tend to have a continuing relationship with a consultancy company, many smaller firms when they run into trouble turn to Management Consultants for help.

What are the major reasons for hiring a Management Consultant

(1) He has a wide range of knowledge of business operations often covering many different industries. This insight he has makes it easier to understand your problems.

(2) He has experience in dealing with computer specialists such as equipment suppliers, software houses and computer bureaus and can cut through the jargon to get to the real facts that need to be communicated.

(3) In dealing with any client problem he uses a professional approach which helps him deal with the problem in logical steps and also assists the client to understand what is happening.

(4) Part of his time is devoted to keeping up to date with technological changes. In the computer industry this is particularly important as the rate of change tends to be faster than in other industries.

(5) He is an experienced agent of change and comes from outside the company's political arena. Many users in conflict with a data processing department will be more forthcoming in dealing with an independent consultant with the result that change is easier to achieve and results are longer lasting.

What is the Professional Approach used by Management Consultants?

The first problem faced by any Manager who is confronted with the symptoms of a problem is determining their cause. The Management Consultant has highly developed investigatory and analytical skills which he can bring to bear to investigate the problem and define it in such a way that a specific assignment can be undertaken to address the causes of the problem.

Next he would want to discuss the scope of the assignment with the General Manager, or person responsible for hiring him, to get agreement on the scale and scope of the assignment.

The next step is the preparation of terms of reference which act as guidelines to ensure that the project stays within the defined scope.

Based on the terms of reference specific activities are identified as needing to be done. The effort and time scale required for completion of each activity is set out formally in a work plan. This work plan is then used to monitor performance against activities.

Project control techniques are an important feature of the professional approach to assignments. They assist communication between the people performing various tasks and the project manager and from him back to the client.

Regular reporting of achievements and problems to the client's senior management is an important feature of effective communication. A well informed client is a happy client.

In order to ensure that the fruits of his efforts continue to be available for communication after he has completed the assignment the Management Consultant employs comprehensive documentation techniques. Documentation extends from records of discussions, through reports containing recommendations, to system specifications and procedure manuals.

Management Consulting for Data Processing Managers

In New Zealand most EDP Managers tend to follow the Kiwi philosophy of 'do it yourself'. The use of Management Consultants by EDP Managers is still in its infancy. Why is this so? Other professions recognise areas of specialisation within their profession calling for referral of specific work to specialists. In medicine the existence of anaethesists, radiologists and surgeons has been recognised for a long time. General practitioners often refer patients to skilled specialists without any fear of losing the patient as a client.

Architects designing buildings refer structural work to engineers and estimating of materials to quantity surveyors.

Perhaps because data processing is still going through birth pangs as a profession there still exists a fear of being found lacking in some way. Also many DP Managers have come from technical backgrounds where they have acted as professional advisors on technical matters. Suddenly they are thrust into a position calling for the exercise of management skills. One of the most important management skills is achieving results through others. Another skill is being able to admit a lack of knowledge or recognising when performance is less than desirable. Without this initial recognition, managerial action to improve the situation is impossible.

Which are the fields where a DP Manager may best seek help from a Management Consultant?

Here is a selection:

(1) Evaluation of an Organisation's Structure relative to defined organisational objectives. (Are the objectives defined in a way which makes them realisable?)

(2) Review of computer operations. How are we doing relative to other DP organisations? How good are our Standards (if we have any)? How effeciently are we using our human resources and our equipment.

(3) Review of systems. Are we doing the right things? Are we doing things right?

(4) Review of equipment. What performance are we getting? How can throughput be increased. What do we require to meet our processing needs in 5 years time?

(5) Preparation of systems planning studies. How do we present to management a plan which clearly identifies the systems which offer the best cost benefit results from the organisation's point of view (as opposed to representing a sectional interest)?

(6) The economic evaluation of a specific function or functions within an organisation. Where does our revenue really come from? Do our controls really help to keep costs down or just record their occurrence? Where do the real sources of costs lie?

(7) Equipment Selection. How can we choose, when each manufacturer's proposal stresses different advantages? How do we relate the performance of a super disc against a whizz microfilm unit. How do any of the items of hardware proposed relate to my company's real needs for information processing? How do I know the systems that go with the equipment will actually give me the kind of result I want?

(8) Review of Administrative Procedures. What reporting system do I have? Why do we keep running out of forms? Why do I keep losing people? (Do we have a personal evaluation programme and regular salary reviews?) How do we know when peaks and backlogs occur?

(9) Definition of systems requirements. How do we get the user to tell us what he really wants? How do we communicate with the users when they have their jargon and we have ours?

(10) Recruitment of key personnel. How do you find the sort of person with the required knowledge and skills who will fit into your organisation? Have the people we recruited contributed as much as we would have liked? Did they stay?

Who gets the benefits of hiring a Management Consultant?

(1) **The Management Consultant**

Obviously the Management Consultant gains. He is paid a fee for his services and hopefully gets the satisfaction of doing a good job.

(2) **The Hiring Organisation**

- Internal communications become more effective. The project control techniques and regular communication techniques used by the Management Consultant generally have the effect of opening up channels of communication especially where these have been clogged, and of introducing new communication techniques.

- External communications become more effective. The Consultant helps the organisation to define what it wants more clearly.

- Achievement of project goals at pre-determined cost. How many DP Managers know the real costs of developing and implementing a new system? By using the professional approach the Management Consultant is able to assist management to achieve goals at a cost which has at least some resemblance to the original estimate.

- Greater certainty that the problem solutions is long term and maintainable. Most users want solutions to their immediate problems. In the long term this leads to a creeping and unmanageable DP commitment. Properly defined user requirements ensure that all real requirements are defined and plans can be developed to meet them in progressive stages if necessary. Identification of variables that are likely to change enable them to be included as maintainable table files rather than built into programs.

- Less management time required to develop the solution. Management from the hiring organisation would be consulted frequently during the assignment, but the larger scale commitment of effort, if the organisation attempts its own solution, is saved.

(3) Computer Equipment and Software Suppliers

Computer equipment and software suppliers stand to benefit too when they deal with an organisation through a Management Consultant. They benefit in the following ways:

(1) They generally get a clearer definition of the customer's requirements.

(2) There is greater assurance that the client organisation's management understands what they are buying.

(3) There should be a better definition of the time frame for achievement of equipment or software implementation relative to the client organisation's other activities.

(4) They know that selection is based on a 'best fit' basis rather than on sales techniques and customer 'understanding' of what is being delivered.

Traps to Avoid

In hiring a Management Consultant there are traps which if avoided lead to a better client - consultant relationship by making sure that the expectations of each are clearly defined and understood. These are some of the traps to avoid:

- Engaging a consultant without defining terms of reference.
- Using the consultant as a general advisor.
- Seeking assistance on problems outside the terms of reference (unless they are extended) - the creeping commitment.
- Using the consultant on day to day activities (unless hired as a facilities manager).
- Not making sure key people make time to be consulted
- Not differentiating between 'elapsed time' and on the job time
- Hiring over-committed consultants
- Paying unnecessary retainers
- Delaying decisions which are needed for the assignment to proceed
- Frequent changes of mind - it extends the assignment
- Blaming consultants for everything that goes wrong afterwards. (You cannot expect presents to arrive at Christmas unless you post them on time).

Believing your own problem is unique. Technical processes may certainly be unique but similar management problems keep recurring in all organisations.

What are the costs of Hiring a Management Consultant?

When you hire a consultant you will pay fees and expenses. Methods of determining the actual fees charged vary. However, the basic unit for charging time is the consultant's hourly or daily rate. The actual rate varies depending on the consultant's reputation and experience.

Expenses are charged in addition to fees. Expenses consist of travel and accommodation expenses plus disbursements for services such as typing, photo copying and presentation material.

Contracts with consultants fall into three categories:-

(1) The fixed ceiling contract
(2) The fixed price contract
(3) The retainer contract

The fixed ceiling contract sets the upper limit to fees within which the consultant carries out the defined activities for his client.

The fixed price contract may be all inclusive (i.e. cover expenses also) or just relate to fees. This defines the amount payable, by the client for performance of activities by the consultant regardless of the actual hours taken. The risk is greater with fixed price contracts so the charge is likely to be higher, also the activities are likely to be far more precisely defined to reduce the chances of non fulfilment by the consultant.

The retainer contract is a means of attempting to get priority attention for the client in the event of need. It usually consists of a fixed base price which you pay whether you get any services or not. This base price should cover a specified number of consultant time units which can be taken at any time during the year. Units above those specified are then chargeable at the normal rate. A variant on this is a lower base price which covers no services but is intended to assure prompt attention when the need does arise.

What are the Potential Savings of Hiring a Management Consultant?

Given that organisations need the help of consultants for three main reasons: Their expertise, to get an independent view and to achieve results at lower cost than by using the organisation's own resources, where do the savings lie?

- Avoidance of recruitment costs
- Avoidance of training costs
- Avoidance of termination costs
- Avoidance of peak loading (overtime costs).

Summary

A management consultant uses effective communication techniques:

- To bring about change smoothly and with as little pain as possible
- To ensure the solution matches the problems as closely as the client's resources permit
- To document the achievements fully so that they can be maintained in the future.

SPECIFICATION TO IMPLEMENTATION IN 24 HOURS

AUTHOR: K.I. MITCHELL, M.Sc., M.N.Z.C.S.

Computer Consultant,
J.P. Scott & Associates Ltd.

ABSTRACT:

The Analysis Process for the small first-time commercial user is reviewed and the steps required in the process elaborated. A system for documenting this phase is explained. Some common program structures are identified.

A software system can then generate on-line file maintenance and enquiry facilities, permit posting of batch data and provide audit and log facilities. File listing utilities are immediately available. Complementary reporting programs are generated using a simple procedural language which is self-documenting with the data-names familiar to the user through the prompts of the data entry facility.

Supporting documentation required includes the cost benefit study, why certain business decisions were made and external document flow.

The role of the applications programmer is eliminated.

Extensions into networks and "paging" distributed databases are discussed.

KEYWORDS:

User, Hardware, Analysis Process, Software, Incremental Compiler, Field definitions, Record descriptions, Screen Formatting File Organization, Logfile, Report Generator, Sort, Utilities.

C.R. CATEGORIES

3.50, 4.12, 4.13, 6.20

The User:

The greater proportion of business in N.Z. is conducted by small businesses of less than 75 employees. They typically have small debtors files (less than 1000 debtors), small accounts payable (less than 200 regular creditors), small general ledgers (less than 200 accounts) and depending on the nature of the business relatively small stock files (less than 1000 items) but the inwards goods and transportation-related areas can be substantial problems due to overseas suppliers, currency fluctuations, irregular deliveries, back-ordering and geographical distribution of their market. However the number of invoice lines per day may be over 1000 requiring some mechanisation.

Such businesses are frequently run by the owner employing few staff and typically the key person is the office girl. When she leaves or comes under pressure due to growth, a fully installed computer system at under $20,000 purchase is attractive to the owner.

The owner is usually very well informed about his business; has clear cut ideas on what he wants; feels vulnerable to the quality or otherwise of his employee's work; and is very concerned about the peculiar structure of the EDP industry and its very complicated quotations for hardware and software. He realises that it is quite contrary to his objectives to employ EDP staff. Computerisation is desirable - how to proceed?

The Hardware:
While the system outlined in this paper could be implemented on larger machines it is targeted on the smaller user.

A typical file of 500-5000 records of 50-500 bytes per record requires from .25 to 2.5 Mbytes of logically contigous direct access storage, i.e. floppy disks are big enough.

Allowing 4 bytes per record for tags and 4 to 12 bytes for keys, 500-5000 records would require from 4k to 80k to sort.

Suitable configurations can be purchased from as low as $14,000 including tax.

The development cost must not be out of proportion or the economy of the technology will be lost.

The Analysis Process:

The analysis process begins with a discussion on the objectives of the business, clarification of the target market, strategies and hence a business plan. From this is crystalised the requirements of the user for information for decision-making and externally imposed essential essential paper work.

Following a survey of existing approaches (during this phase volumes are established) tentative report layouts are drawn up to firm up discussion and the use of each report and the purpose of each data element is reviewed. Such details as how current each item of data must be to be of value are checked. The benefit of a computer system can now be evaluated.

The consultant now reviews all available inputs, eliminates unnecessary items, ensures adequate checks are available and plans the data entry function.

Placing great weight on the KISS (Keep it Simple System) the file structures are planned and data items listed for each record or database element. A brief functional description is given for each process and finally a list of all required programs (or modules) can be made.

Sufficient detail is now available to cost the computer system, i.e. keying volumes, printing volumes, file sizes and development details permit the costing of hardware, software and operation. The final cost-benefit study can be completed and the user make a decision to proceed.

The program development process may now begin.

System Facilities:

In commercial systems the following facilities are desirable:

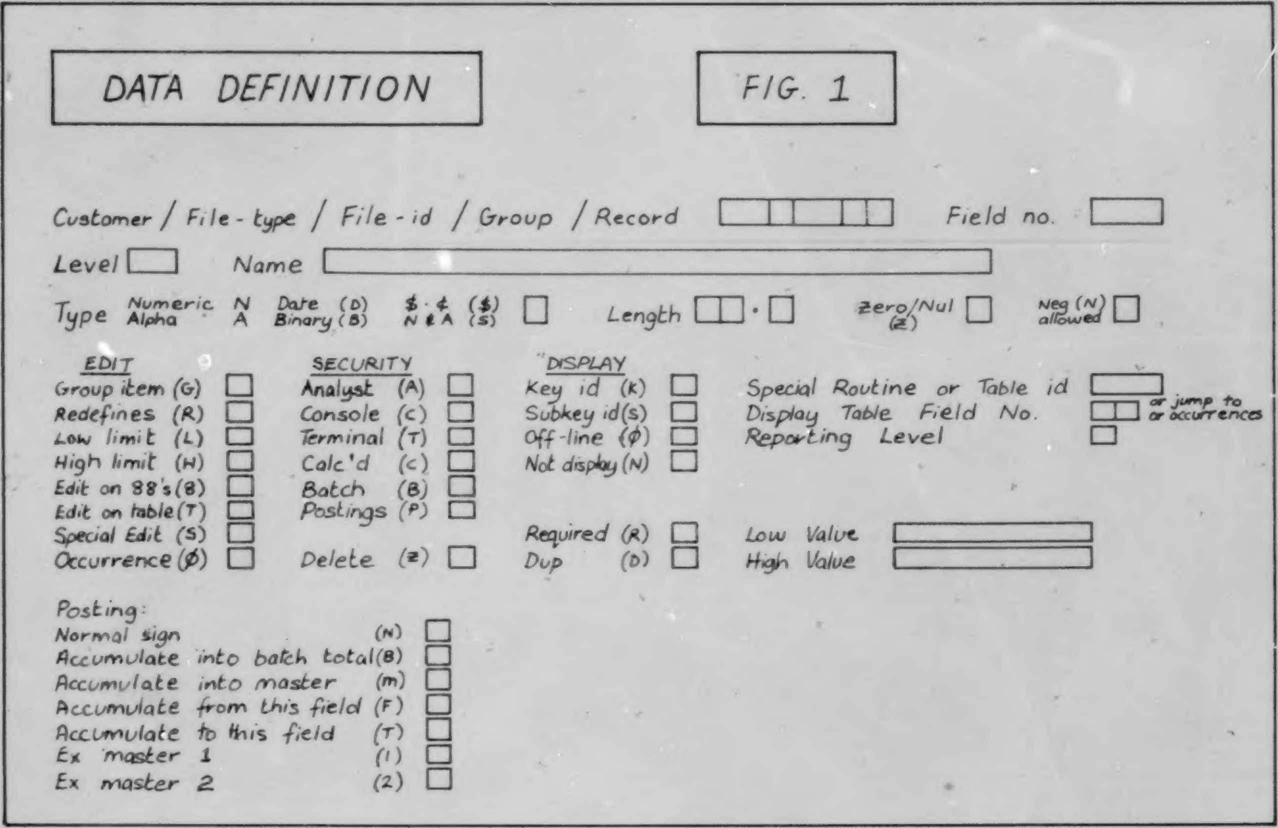
a. On-time maintenance of master file records, showing prompts, fully editing the data
b. On-line enquiry
c. On-line keying and editing of batched transactions which directly update the master file.
d. File and log listings without further specification
e. Easy-to-use report generators
f. Utility backup and recovery
i. Low visibility operating system
j. Suitable file organisations
h. Adequate control structures
l. Batch program capability

Specification:

Early in the analysis process data items are identified and a tentative classification into records and files made. These details are entered on a form (Figure 1). Attributes identified at this time are:

Name (meaningful to user)
Type - Numeric, Alpha, Date, Binary
Length and decimal position
Whether sign allowed or zero or NUL allowed
Editing details including special routines and table edits, absolute limits.

The analyst now enters these details into the machine via a VDU. They are fully edited. During the keying he may make minor decisions on the other attributes and controls over the data. Changes are easily made. Typing delays are eliminated. A clean printout showing



DATA DEFINITION

FIG. 1

Customer / File-type / File-id / Group / Record ☐☐☐☐☐ Field no. ☐

Level ☐ Name ☐

Type Numeric N Alpha A Date (D) Binary (B) $.¢ ($) N & A (S) ☐ Length ☐☐.☐ Zero/Nul (Z) ☐ Neg (N) allowed ☐

EDIT
- Group item (G) ☐
- Redefines (R) ☐
- Low limit (L) ☐
- High limit (H) ☐
- Edit on 88's (8) ☐
- Edit on table (T) ☐
- Special Edit (S) ☐
- Occurrence (0) ☐

SECURITY
- Analyst (A) ☐
- Console (C) ☐
- Terminal (T) ☐
- Calc'd (C) ☐
- Batch (B) ☐
- Postings (P) ☐
- Delete (Z) ☐

DISPLAY
- Key id (K) ☐
- Subkey id (S) ☐
- Off-line (0) ☐
- Not display (N) ☐
- Required (R) ☐
- Dup (D) ☐

Special Routine or Table id ☐

Display Table Field No. ☐☐ or jump to or occurrences

Reporting Level ☐

Low Value ☐

High Value ☐

Posting:
- Normal sign (N) ☐
- Accumulate into batch total (B) ☐
- Accumulate into master (M) ☐
- Accumulate from this field (F) ☐
- Accumulate to this field (T) ☐
- Ex master 1 (1) ☐
- Ex master 2 (2) ☐

415

Data Definition Printout   Figure 2

```
A B C LIMITED   MM-DEBTORS MASTER              RECORD: MASTER
04    CUSTOMER  NAME                 ALPHA   37 NUL NOT ALLOWED

EDIT - NONE
SECURITY - ANY TERMINAL
DISPLAY - POSITIVE ID
-----------------------------------------------------------------
A B C LIMITED   MM - DEBTORS MASTER            RECORD: MASTER
18  90& DAYS                                   $8
SECURITY - POSTINGS ONLY
         - MUST BE ZERO BEFORE DELETION
-----------------------------------------------------------------

A B C LIMITED   MM -  DEBTORS MASTER           RECORD:  MASTER
20 CREDIT STATUS                               A1
EDIT ON CONDITION NAMES      STOPPED S
                             REFER   R
                             CREDIT_OK -
                             IN_COLLECTION C

SECURITY CONSOLE ONLY
-----------------------------------------------------------------

A B C LIMITED   MM  -  DEBTORS MASTER       RECORD:  MASTER
15 BRANCH                                  N2 ZERO NOT ALLOWED
                                             NEGATIVE NOT ALLOWED

EDIT ON TABLE       TABLE ID BRA           BRANCH NAMES

                    DISPLAY FIELD 5        BRANCH NAMES
-----------------------------------------------------------------
A B C LIMITED   ST  STOCK                  RECORD:  HISTORY
19 ORDERS                                  N4 NEGATIVE NOT ALLOWED
SECURITY BATCH ONLY
OCCURS 13
* BUILT BY MONTHLY STOCK ANALYSIS REPORT
-----------------------------------------------------------------
```

1 data element per page is available for filing in the clients set of documentation where it -may be reviewed (Figure 2).

The final decisions are made on record formats.

For the benefit of the client the analyst may now key some data demonstrating the sequence of keying that will be required and showing the enquiry facilities. The natural listing format of the file may now be printed off and form a basis for discussion of the tailored reports.

Most reports require selected fields from a file to be listed perhaps truncated or rounded, possible control breaks, with additional totals accumulated. A report of this nature may be quickly keyed through the report generator, a sample printed off and shown to the client. Note that tedious printer layout charts are not required as the incremental compiler will point out if the lines are too long and will permit truncation of the print line. It is easier to maintain the code than a printer layout chart.

At this point if the client is close to a final decision his staff may be trained and master file creation commenced. Other parts of the documentation may be keyed and maintained through Text Editors and RUN-OFF (DEC).

There remains only the problem of specifying the few remaining programs which are more complex than file listings or transaction processings. However, they are few. Typically there are none in debtors or creditors. An invoicing program if invoices are printed as keyed, requires the following:

Specify name and position of fields keyed on the header.
Specify literals and where to print them
Specify where masterfile fields are to be displayed.

Specify line item extension and accumulation and back order alternatives.

Specify total display including freight and discount calculations.

Note that if the line items are keyed as transactions the invoicing may be specified as a batch report.

The batch report facility provides file matching and control break logic.

Stock systems require some forecasting routines. Simple exponential smoothing is provided as a verb on fields specified as "OCCURS DEPENDING ON".

Sales Analyses require summaries and typically make use of table files to hold literals. Accounting entries also make use of the table files to hold account codes and literals. Table file information must be supplied by the client - by recognising its special nature the system provides automatic printout and maintenance eliminating the need for later program maintenance.

Dut to the nature of the incremental compiler it is quicker to key the program than write a specification and the code is highly self-documenting and is readable by the client. Within 24 hours of the last report being specified by the analyst with the client the system may be handed over to the client. It is already fully documented.

Using this system a competent analyst may require from 40 to 100 hours to implement a system of debtors, creditors, general ledger and stock control. At lease 50% of this time is spent with the client. The analysis and design process is highly visible to the client greatly increasing his confidence. Communication with the client is direct, minimising misunderstandings.

The client is not stimulated into asking for everything for nothing and can readily see the amount of work required to add frills and hence has a better appreciation of the cost justification or otherwise of marginal facilities. The code is "bullet-proofed" against changes in record layout. The system lends itself to using existing file descriptions for new clients.

Techniques:

a. Self-defining data
b. Field definitions
c. Record Descriptors
d. Screen Formatting
e. File organisation and log file
f. Postings and Batch Control
g. Special Edits
h. Self-reports
i. Report Generator
j. Sort
k. Utilities.

Self-Defining Data:

Every data item is self-defining - carrying two bits indication:

a. an integer 0-999
b. a number in the range ± 9(8)
c. a number in the range ± 9(13)
d. an alphanumeric field

An alphanumeric field may be from 1 to 63 bytes long or it may be a set of 8 binary switches.

Hence the length of every field is known and a record and hence a file may be listed in a fully formatted form without further knowledge. The major reason for this design decision was that the decreasing cost of direct access storage has led to greater use of alphabetic data items which are inherently variable length. The second reason is that the compiler/interpreter logic can be simpler and "bullet-proofed" against field changes. Compiled programs are impervious to file changes other than deletions.

Field Definitions:

Each field has a descriptor consisting of not less than 5 bytes plus its name. This descriptor is available to the VDU interpreter and to the compiler. It may contain any of the following:

Data Type - Numeric, digits to left, right of decimal point
$.¢, digits to left of decimal point
Alphameric, maximum length
Dates, from 1975 or earlier
Binary switch

Edit - negative allowed, zero allowed limits
limited set of values (COBOL 88 levels)
against a table independently maintained
special routine

Security - levels
calculated only
maintained by batch processing only

Keying - required
must be zero on delete

etcetera.

If a posting then there may be:

field completed from master
accumulate to batch control
accumulate into master
sign dependent on record code

etcetera.

Record Descriptions:

A record consists of a set of fields. It has a name, belongs to a file, has a variable length and contains security codes on deletions and insertions. Every record contains file identification, and a fixed portion giving time-stamp, sequence of postings, times accessed and binary switches. An "OCCURS DEPENDING ON" facility is provided.

Files are identified as:
- system files
- master files
- backup (physical)
- sequential (sorted)
- log
- chained postings
- and sequential (sorted) postings.

The sequence of key fields is fixed although the keys themselves are variable.

Non-standard randomisation algorithms and SOUNDEX key manipulation may be plugged in. Cross-relationship edits may be coded.

Screen Formatting:

The screen used has format-protection. Basically a menu routine selects a record description. A code for add/change/delete is entered, together with the key fields. Identifiers are displayed on change/delete. A prompt is displayed showing the name of the data item and the number of positions which may be keyed. Data items are keyed singly with carriage return. The operating system allows "key ahead". Fields are edited and give standard error messages clearing the remaining portions of the screen. A positive response is required to clear errors whereupon the screen is reformatted from the error field. A control character can suppress prompts in which case the positive ID fields are displayed at the end or on a different control character. This later option is important for volume data entry giving more time for overlap of keying, editing and accessing. Field descriptors permit alternative display layouts. On changes a control character permits full displays of fields and the cursor may be positioned to the field to be changed which is cleared to end of line and keyed.

File Organisation and Log File:

In any on-line updating system every attempt must be made to access a masterfile record in one head movement. The simplest approach is a randomised disk file. The overflow is handled by chaining (forwards only) from the prime block. New records are inserted in logical sequence in the prime block and its chain. This sequence of short chains is used in the sort algorithm.

Postings may be chained from their master. They are inserted into the prime block with forward chaining until or unless that block overflows. They are then loaded into an overflow area with a forward chain; the master has pointers to the first and last of chain. Hence a posting will update three logical blocks; for itself, its master, and a chain. Alternatively postings may simply be thrown forward into the log file.

The log file is forced to disk on every change of record type or end of batch or quiescent point in the processing. It carries pointers to the last block assigned from the overflow pool enabling recovery after failure. All maintenance changes are logged.

At the end of day the log file is printed as a self-report as the format of every record is known.

Postings:

Batch input may be keyed. Batch control records are assigned and posted into the database at beginning and end-of-batch. A completeness-of-input batch report may be taken at any time. Batch input transactions may reference and display ID's from one or more master records and accumulate into them as well as batch totals. Batch numbers are assigned to prime batches but enterd for corrections which use the same batch records.

Special Edits:

Special edits mainly of the cross-relationship type may be coded. The code generates sub-routines which are loaded dynamically. They may be invoked on keying a field or at end of record. The output may be an error message and rejection of field update or calculated fields. These routines are automatically invoked on a file validation pass.

Self-Report:

Because the form of every field is known it is possible to print out every field in a fully formatted form, complete with headings and totals. Input validation routines may be re-applied. This is a powerful facility enabling the client to see every data item without reference to specialised staff, confirming his understanding of how fields are derived.

Report Generator:

The Report Generator is an interactive compiler. At compile time the record formats are available to it. All verbs may be keyed in short forms and all datanames with the minimum characters required for uniqueness. After each carriage return the line is cleared, re-echoed in full and indented appropriately. the DUP key is available. The code is generated incrementally and hence execution is immediate. A reasonable report can be keyed in about 500 keystrokes. An example of such a report is shown (Figure 3)

THE PROGRAM AS ECHOED OR PRINTED.

```
 1 PROGRAM:UXCUTB
 2 HISTORY:
 3          VERSION 01.00 16JUN78 3.30PM K. I. MITCHELL
 4 TITLE: TRIAL BALANCE:MONTH-END:132
 5 FILE-ID: CU M DEBTORS MASTER
 6 SEQUENCE:CUSTOMER NO
 7 RECORD: MM DEBTORS MASTER
 8        DEFINE:TOTAL $8.2
 9        DEFINE:OLD-BAL $8.2
10        DEFINE:COLLECTION N2
11        COMPUTE:TOTAL=90&-DAYS+60-DAYS+30-DAYS+CURRENT
12        LINE:
13              SPI:2
14              CUSTOMER-NO
15              CUSTOMER-NAME
16              90&-DAYS +
17              60-DAYS +
18              30-DAYS +
19              CURRENT +
20              TOTAL +@
21              TELEPHONE
22              LAST-PAY-DATE IF 90&-DAYS>0 OR 60-DAYS>0
23              LAST-PAY-$ IF 90&-DAYS>0 OR 60-DAYS>0
24              LAST-INVOICE if 90&-DAYS>0 OR 60-DAYS>0
25        LINE:
26              ONLY IF STOPPED
27              '**STOPPED**' AT CUSTOMER NAME
28        LINE:
29              ONLY IF REFER
30              '**REFER**' AT CUSTOMER NAME
31        LINE:
32              ONLY IF LAST-CALL<6-MONTHS-OLD
33              LAST-CALL #AT CUSTOMER-NAME
```

```
      ACCUM:
            90&-DAYS
            60-DAYS
            30-DAYS
            CURRENT
            TOTAL
            CASH-LAST-MONTH
      COUNT:
            TOTAL
AT-END:
      DEFINE: COLLECTION N3
      COMPUTE:COLLECTION=90&-DAYS+60-DAYS+30-DAYS
      COMPUTE:COLLECTION=CASH-LAST-MONTH*100/COLLECTION
      LINE:
            SKI:3
            COLLECTION #AT CUSTOMER-NAME
            '%'         NOSPACE
            'TOTAL' AT 90&-DAYS - 6
            90&-DAYS
            60-DAYS
            30-DAYS
            CURRENT
            TOTAL
            'NO. OF CUSTOMERS' AT LAST-PAY-DATE
            @ TOTAL
```

THE PROGRAM AS KEYED.

```
 1 UXCUTB<>
 2
 3
 4
 5 T TRIAL BALANCE:M<>
 6 F CU M<>
 7 S<>
 8 R<>
 9 D TOTAL $8.2<>
10 D OLD_BAL $8.2<>
11 D COLLECTION N2<>
12 C TO=90&+60_+30_+CUR<>
13 L<>
14 SPI2<>
15 CUSTOMER_NO<>
16 CUSTOMER_NA<>
17 90& +<>
18 60_ +<>
19 30_ +<>
20 TO +@<>
21 TE<>
22 LAST_PAY_D IF 90&>0 OR 60_>o<>
23 LAST_PAY_$ (dup)
24 LAST_I (dup)
25 L<>
26 O ST<>
27 '**STOPPED**' AT CUSTOMER_NA<>
28 L<>
29 O R<>
30 '**REFER**' AT CUSTOMER_NA<>
31 1<>
32 O LAST_C 6_M<>
33 LAST_C #AT CUSTOMER_NA<>
```

Cont'd.

```
34 A<>
35 CA<>
36 AT<>
37 D COLLECTION N3<>
38 C CO=90&+60_+30_<>
39 C CO=CA*100/CO<>
40 L<>
41 SKI3<>
42 CO #AT CUSTOMER_NA<>
43 '%' NOSP<>
44 'TOTAL' AT 90&-6<>
45 90&<>
46 60_<>
47 30_<>
48 CUR<>
49 TO<>
50 'NO OF CUSTOMERS' AT LAST-PAY-D<>
51 @TO<>
```

Files may be matched, sorted and control breaks detected. Files may also be updated either serially or in-place on the randomised file.

Sort:

A tag sort has been developed. It handles the variable length fields, sorts in core by connecting the keys into chains and writes the gags out in sequence into a separate file at end. It is automatically invoked by the Report Generator. The reaccessing of the file is fully overlapped with printing.

Utilities:

A full set of utilities to copy files physically or logically are available and files may be readily reformatted or created from others. These include back-ups and recovery facilities.

Extensions:

A compiler orientated towards the VDU to enable interactive processing is being specified. Essentially it is an extension permitting the display and entry of posting fields, masterfile fields, literals, error responses, generated items and calculated results on the screen.

The system may be placed on line to a host by adding two tasks - one to trickle the log file back to the host; the other to request the host to send masterfile images down to the remote which will then build them into its own randomised files. When the remote database is getting full a "Garbage collector" program sweeps the database in physical sequence deleting inactive masters. Hence the remote database contains a subset of the host database selected by FIFO, i.e. a paged distributed database. The record descriptors may permit the host to delete certain fields before transmission thereby reducing remote record size. The remote database is redundant except when communications are cut.

SOME THOUGHTS ON COMPUTERS AND LANGUAGE

AUTHOR: PATRICK T. NORRIS,
has worked for IBM for 14 years, as
a Programmer, Manager and more
recently as a Marketing Representative.

ACKNOWLEDGEMENTS:

Grateful acknowledgement to my wife, Janet, without whose knowledge of linguistics I would have foundered completely.

ABSTRACT:

This paper reviews the trend of recent developments in the area of man/machine communication, using STAIRS, IIS and APL as examples. The problems associated with the machine translation of natural language, and future implications are also explored.

KEY WORDS AND PHRASES

Language, computer assisted instruction, CAI, text retrieval, STAIRS, APL, translation, linguistics, IIS.

C.R. CATEGORIES

2.12, 3.32, 3.42, 3.71, 3.74, 3.75, 4.13, 4.19

(A) A few years ago, in the dawn of computing, it was confidently predicted that computers would soon hold conversations with humans, would translate French into English, American into Russian, displace teachers from schools and most favoured of all predictions:
.... "a computer would win the world chess championship".

This early promise has not been fulfilled, but where are we at in the use of computers with respect to ordinary language?

In this paper, the author explores three uses of computers in this context and presents some speculative views on the relationship between computers and language.

The three uses of computers explored for illustrative purposes are:

(1) A text based retrieval system - The Storage and Information Retrieval System (STAIRS).

(2) A computer based training system - The Interactive Instructional System (IIS).

(3) A personal computing system - A Programming Language (APL).

(B) (1) STAIRS

STAIRS is a system for the interactive retrieval of documents dependent upon certain specified search criteria. The documents are first loaded into a database and at the same time indexed by every word contained in the documents. Certain word or abbreviations may be ignored for indexing purposes (e.g. the common prepositions and articles).

Subsequently, when an enquirer specifies a word, STAIRS presents all documents containing that word. The enquirer may then refine his query by specifying further words if desired.

Complex queries may be asked of the system - facilitated by a range of Boolean operators (AND, OR, XOR, NOT) plus some additional STAIRS operations - ADJ for adjacency etc (e.g. "North ADJ Island" would retrieve only documents where the words North and Island occurred adjacent to each other).

Previous queries are stored and may be displayed or used in subsequent queries.

An enquirer can specify a prefix only if desired, the system presenting all words beginning with that prefix.

Textual data has characteristics rather different from the numerical data with which most of us are familiar. It is usually much more untidy. People spell differently, different abbreviations abound, typists make errors and most important of all, most living developed languages are very rich in symonyms: Should an enquirer specify "CAR, AUTOMOBILE, AUTO, HOT-ROD, MOTOR VEHICLE" or all of the above? Clearly it would depend on the enquirer's perception of what the database was likely to contain, its source and age. It would be of little use searching for "COLOUR" in an American database or for "TRANSISTOR" in the Bible.

It is important to realise that the computer system is regarding the words only as character strings - it is not ascribing an intrinsic meaning to them. If an enquiry is made for "SECOND" then the system will retrieve all occurrences irrespective of whether the meaning is:

(a) 1/60th of a minute
(b) An assistant at a duel
(c) Attach to another unit or department
(d) Next after first

and it is often necessary to exclude undesired selections of this nature by further refinements.

This represents an aspect of linking the strengths of the human mind and the computer. The computer is very fast but very stupid; it very quickly retrieves all occurences of the character string "SECOND" but is unable to discern different meanings in context. On the other hand the human mind can tell in an instant which of the documents selected are relevant and which are superfluous.

When demonstrating this system I usually invite the audience to present their own word to the database (a collection of abstracts of US Government Departmental Reports). Invariably, someone yells out "SEX!"; the word is typed in and back comes the selection - a scientific report on determination of the age and sex of frogs in North Wisconsin! Perhaps we should have typed "SEX!!"

To help enquirers cope with the richness of a living language, a thesaurus capability has been added to STAIRS. This facilitates the broadening of a query to include a wide range of synonyms together with the ability to recognise alternative conjugations of verbs (e.g. go, going, went, gone). This thesaurus capability is currently available in six languages, (in case one should feel the urge to browse (forage, track, grub) through a few foreign databases).

Text base retrieval systems are here to stay and we will see rapid expansion of their use as the number of terminals increases and the cost of storage continues to tumble. At $10 per megabyte it is as cheap to store a book on a mass storage system as buy two extra copies (ignoring transcription costs). Further the linking of databases through telecommunications will greatly enlarge the scale and recency of information available to a single user.

(2) The Interactive Instructional System - IIS

This system for computer assisted instruction - CAI, allows an author to construct courses building in questions designed to test the students knowledge and understanding of the relevant material. Based on the replies, a student may be allowed to proceed or perhaps be directed to remedial learning. The approach is based on the concept of "Programmed Instruction" or PI and will be familiar to most of those who have undergone initial DP training in recent years. (I was inducted in the old steam instructor days where the only aid was a template and the testing was done on butchers paper).

The system also keeps track of student progress and allows a student to resume from where he left off at a previous session.

The instructor has access to student responses and may use them to determine problem areas. In addition the instructor and students may exchange messages. These exchanges may take place live or be left awaiting the other's attention.

This aspect of the training system is rather similar to the "laboratory" approach to the teaching of modern languages, where the tutor monitors the student's progress and may choose to intervene to assist or clarify material as required, but in general letting each student proceed at his own pace.

The success or otherwise of a computer based training system depends very heavily on the quality of the course material and whether it meets the student needs.

Some subjects are much more readily taught through this medium than others. IBM first developed computer based training for its Customer Engineering force as being the most effective way of disseminating formal training in the maintenance and repair of IBM equipment. This has proved successful and indeed this experience forms the basis of IIS and similar products.

The computer based training clearly has limitations where the subject matter is more abstract in nature. For instance the response to:

"Martin Luther's impact on 16th Century European thought was comparable to that of Chairman Mao on 20th Century Asia. Please discuss"

is certainly beyond the scope of IIS to assess. (In a trivial sense of course, the system could accept such essay responses and route them to an instructor for marking).

Computer Assisted Instruction has many detractors and it is beyond the scope of this paper to put the general case for an against except to say that in the case of data processing training especially we are already seeing steady growth of this technique in parallel with video and traditional classroom training.

(3) <u>APL</u>

Although APL can properly be regarded as a fully fledged programming language, in the main its devotees have been people who were not professional programmers, but usually engineers, scientist, economists, statisticians, financial planners and analysts. The appeal is quite simple. This group typically is looking for a tool to handle data arrays in as simple and logical manner as possible.

APL does just that through its inherent simplicity and concise notation. How beautifully elegant and simple to say

```
          A              B + C
```

and know that the same statement will work whether A, B and C are scalars, rows or even an n-dimensional array and that that decision need only be made as the statement is being executed.

Remember all that verbosity of COBOL and PL/I and all those bureaucratic constraints on subscripting. APL has a wonderfully rich range of operators although, of course, the character set has been the cause of much of the prejudice against it.

"All those Greek letters reminds me too much of Maths III".

(A statement made by the author prior to being hooked).

A more practical objection was the demands made to alter keyboards, golfballs, VDU's and the like and the constant software hassles over character set compability with other more prosaic languages.

Despite these constraints, its growth in usage has been astronomical and it is well on the way to becoming the number one language for non-DP professionals. Adding to this class of users is a further group who use APL indirectly, without necessarily realising it. Because APL is so good for writing interactive programs it is widely used for question and answer, step by step, problem solving. A notable example of this approach is the IBM internal "AIDS" system which is provided to assist field personnel in the development of precise equipment configurations.

Because of the structure of APL it is easy to create one's own functions (verbs) and variables (nouns) and thereby build up a micro language. These in turn may be combined to form more complex structures.

PLOT A BY B
COUNTRY WHOSE CAPITAL IS ADDIS ABBABA
POPULATION OF FRANCE
LIST ALL OF THE ATTRACTIVE WOMEN

This certainly looks a lot more like idiomatic English then what we've been used to from most computer systems and in my view represents the best hope in a practical sense of making them appear more human.

(C) Let us return to the problem of language translation to which I alluded in my introduction. At first glance it seems an easy matter to translate from one language to another but the analyst however is faced with a much greater problem than simply accessing a two language dictionary.

First, he must cope with homonyms and synonyms. Synonyms are rarely exactly equal in both meaning and their range of appropriate usage. Consider "tall" and "high". One can refer to a building as tall or high and be quite safe but to say a person is high might evoke an altogether different reaction!

Second, there is idiom - the way the words of the language are grouped - these groupings often take on a different meaning to that of the individual component words:

under the influence
"put up with"
"She'll be right"
Je suis vendeur
d'accord
Black black black!
leg before wicket
central processing unit
Kia Kaha
the Birch amendment

The third problem to overcome however could prove the most difficult - how to ascribe meaning. Consider the sentences:

Flying planes are dangerous
Flying plans is dangerous

Although they differ only in that the verb has been changed from plural to singular, the meaning of the sentence is totally altered.

Consider too:

"It is too hot to eat"

This is capable of at least three interpretations:

(a) the weather is too hot for us to eat our meal.
(b) our meal is too hot to eat. (spicey?)
(c) the dog is too hot to eat its meal.

Further examples of interesting or ambiguous sentences:

What did you bring that book
I didn't want to be read to out of up for?

It's good pigs are called pigs because
they are such swine!

The police were ordered to stop drinking at
midnight.

There are also the apocryphal tales of computer translations which converted:

"out of sight, out of mind" to
"invisible idiot"

and

"the spirit is willing, but the flesh is weak"

to

"the whisky is ready, but the meat is raw".

The noted pioneer of modern linguistic analysis, Noam Chomsky, has tried to formulate a grammatical system wherein any sentence may be "transformed" into its "deep structure" from which its meaning could be derived. He pointed the way to a logical and precise analysis of the structure of languages. He hoped to find universals between the deep structures of different languages; he was instinctively striving for von Humboldt's conclusion "that the force which generates language is the same as that which generates thought". Chomsky also stated that linguistic development was essentially creative rather than just a skill re-inforced by invitation. It is thus very much a human attribute rather than mechanistic.

The three approaches to computer interaction do represent an advance over the bad old days and APL in particular does offer some hope for the future.

We are however a long way off having a meaningful high level conversation with a computer (more properly a computer program) in our own language let alone going through the further transformation to other languages.

We must first understand much better that most wonderful of all retrieval and computing systems, the mind of man.

What a wonderful product! - admittedly a long time in development, but:

- compact
- reliable - little requirement for preventive maintenance
- adaptable to new situations
- creative
- audio/visual tactile response unit
- Well proven peripherals (plant installed only)
- standard interface - (AC/DC not type approved)
- OEM's now on the market

As Shakespeare said:

What a piece of work is man
how noble in reason
how infinite in faculty in form, in moving,
how express and admirable!
in action how like an angel
the beauty of the world - the paragon of animals

Bibliography

"Grammar, Frank Palmer", Pelican Books 1971
"The Tyranny of Words", Stuart Chase
"New Horizons In Linguistics", John Lyons, Pelican Books 1970
"Chomsky - Selected Readings", JPB Allen & Paul Van Buren
"Chomsky", John Lyons, Pelican
"Linguistics at Large", Noel Minnis, Paladin
STAIRS/VS General Information Manual, IBM GH12-5114
IIS General Information Manual, IBM GH20-4535
APL General Information, IBM GH20-9064

THE O AND M OF SYSTEMS DEVELOPMENT

AUTHOR: A.J. PAYNE
University of Waikato,
Hamilton.

ABSTRACT:

This paper reviews the phases of systems development and considers the basic actions and tools used. Optimal Conditions for systems development are proposed and a new theoretical model of systems development is presented. The hardware, software and database systems of a "project work bench" is outlined.

KEY WORDS AND PHRASES:

User, Systems analyst, programmer, coding, Data Preparation, Testing, Documentation, Specification, Planning, Communication, Personnel, Project work bench, PERT, Database System, Systems Development.

C.R. CATEGORIES

2.40, 4.0, 4.33, 6.0

THE O AND M OF SYSTEMS DEVELOPMENT

by A.J. Payne
University of Waikato, Hamilton, New Zealand

1 Introduction

This paper presents a system of hardware and software which should comprise the optimal configuration for system development. The first part surveys the phases of system development and reviews the tools and actions which are used in each phase of development. Section two analyses the actions of part one and presents a list of basic actions and tools which are used in systems development. The third section develops optimal conditions based on previous work. Part four presents a new theoretical model of system development based on the optimal conditions which were defined. The new developments portion is then regrouped into sections concerned with hardware, software and database. Finally, some references are given as a basis of library searches.

2 System Development (See Figure 1)

2.1 Introduction

The development of a programming system is well described in Brandon, Palley and O'Reilly (8) and Boehm (17). It starts with an interaction between the user and the systems analyst in the specification of the system. The specification is passed onto the programmer for coding. The coding is presented to data preparation personnel for encoding for the computer. The encoded program is tested by the programmer on the computer. Documentation is prepared in final draft for the users. During the phases, planning is performed and progress/problem reports are passed through the management structure of the organisation. The personnel department aids the project manager with the selection of staff with various attributes for his project. Many of these processes may be recursive and can have two modes. These modes are search for a target corresponding to find 'any' and search for all targets corresponding to find the 'best'.

2.2 Specification

The specification which has been recognised from the earliest stages of computing has had its design methodologies review by Peters and Tripp (18). Teichoew (14) and Davis et al (15) have put forward research studies of the specification of programming systems. Hipo (16) has been used by IBM as a production system but has been found to be more oriented to batch systems.

The techniques used by a systems analyst consist of scanning the problem defined by the user and interpreting the structure in terms of objectives, functions, procedures, input, output and database. The 'design' is massaged according to various philosophies to give a clean, acceptable definition of the system.

Specification can be modelled as a scan of a document from the user against a set of basic rules for the systems analysts. The actions require a scan of the input document for particular facts, a scan of the rules for a particular rule and a scan of the output document to complete the items in particular positions.

2.3 Coding

The coding phase of system development has taken the most study by researchers and this is reflected by a review of twenty-five years of programming languages by Wegner (11). Seegmuller (22) has reviewed the properties of languages and in most cases these properties tend to be more qualitative than quantitive. There have been many attempts to standardise the definition of language. Some definitions have been found to be easy to learn, e.g. Cobol (13) and some have been difficult to understand, e.g. Algol 68 (12).

The techniques used in coding depend on the use of a programming language and manuals describing its operations. The language designers can build in language rules which enhance coding speed, reliability, learning ease and ease of use.

Coding consists theoretically of a scan of the program specification, use of a programming manual and a scan of the coding sheet to position the written program.

2.4 Data Preparation

Data preparation has been studied most frequently from the viewpoint of keyboard design and the subject is summarised by Van Cott and Kincade (19). Ferrara and Nolan (20) have studied the various data entry devices in reference to the cost effectiveness of the systems.

Techniques used for data preparation consist of the use of a normal typewriter keyboard, the stenographic bar board, a menu type light pen system, optical mark reader and magnetic character reader for actual preparation. The variable data uses keyboards and light pens, whereas optical mark and magnetic character characters are used for standard data. Some times voice input is used but mostly in experimental systems.

Data preparation requires a scan of the keyboard and a search of the coding document. Function keys and programmed operations give a method of cutting the amount of work used to produce encoded data.

2.5 Testing

The testing phase takes a considerable portion of any programming system's effort and time. It is described in all its aspects by the books by Hetzel (2) and Brown and Sampson (6). There are two forms of test aid - the unit tester and the system tester.

The techniques include memory and file dumps, traces, result checkers, and subroutine rollback for unit testers. System testers consist of message generators, system exercisors and log files. Much scanning of the program code and test data takes place to determine if, when and where bugs occur.

Testing requires scans of the program for test points, the test aids manual for appropriate test techniques, the test output for correct results and the program for errors.

2.6 Documentation

Documentation is the hate of all computer programmers. Its lack in the systems development cycle was not really appreciated until the NATO Conferences (3,4) of 1969 and 1970. A suggested standard was given by Lomax (1) in 1977.

The techniques available are deciding on the user, defining the subject, and the contents to various levels of detail, the layout, the style and feedback. Sometimes documentation is prepared as files on a computer and run off on a printer of some form.

The documentation phase scans a skeletal set of rules, the program listings, previous documentation and the new manual for positioning the items correctly.

2.7 Planning

Planning in computer projects is starting to be considered as the part which is most under criticism. The chaotic situation of overspent budgets and late projects was appreciated in the NATO Conferences (3,4) of 1969 and 1970. The conferences listed various planning and budgeting techniques to help the manager but these alone have generally failed owing to lack of sufficient data.

The techniques available to the manager listed in Bauer (7) are based on network analysis for each resource in the terms of the quantity which is required at a particular time in reference to the activities of the plan. Progress and problem reports are generated to review the plans which may be modified.

Planning requires a scan of the specification against a set of planning rules to produce a plan as items correctly placed a document. The update process must scan a report, the plan and planning rules to take action on a new plan.

2.8 Communication

Communication has not been a subject of considerable study in the field of system development. It has however been researched greatly in the area of administration and organisation theory. A description of the theory is given by Rogers and Rogers (21).

The techniques used by an organisation in communication consists of written correspondence, phone, conversation and parcel. Some forms of communication are faster for response time than others; some cost more than others; some give better records than others.

Communication requires generating information in the correct order at the correct source in the organisation to go to the correct destination.

2.9 Personnel

Personnel management is an art which is becoming more scientific. Most organisations have policies concerning the selection and control of personnel functions. Methods include position descriptions, aptitude tests, interviews, reviews, objective setting and planning. A review of the processes of personnel management is given in Chandor (5).

The personnel department reviews the organisation searching for job slots, searching for attributes for the job slots from the rules of the organisation and personnel attribute rules and then scanning applicants against the rules of selection to find the employees.

3 Basic Actions and Tools

The theoretical work derived in Payne (23) are an extension of the practical work described by Ivie (24) of a programmer's work bench. It might be called a project work bench. The basic actions of the phases of system development consist of:

(1) scanning an input document for an information item
(2) scanning an output document for an information position
(3) scanning a reference manual/brain for a specific rule
(4) scanning a keyboard/VDU for a key/light pen position
(5) scanning a structure for an element
(6) scanning a structure with multiple criteria.

The basic tools required by the project work bench consist of:

(1) a method for inputing data to the computer
(2) a method for outputing data from the computer
(3) a method of representing a structure in the computer
(4) a method of interpreting input into a structure
(5) a method of interpreting data on the computer
(6) a method of analysing data structures against rules
(7) a method for controlling a structure in the computer
(8) a method for modifying a structure in the computer.

4 Optimal Conditions

The searcher should be experienced especially with the structure and its documentation. He should be a good worker (accurate, efficient, good memory, careful, precise, fast learner) who is able to settle to work quickly and concentrate for long periods. He should use his memory rather than the documentation. If he is forced to use documentation he should have supple joints, long, light fingers which allow pages to slip through them when making a reference. Motions should be kept gentle and within the middle range of movement and concentrated to the fingers only. The searcher should have natural dexterity, aptitude and fast recall and fast analysis. He should adopt the most effective strategy possible.

The structure should be standardised, small, simple, specialised, logically organised, concise, have minimum ambiguity, have minimum error cases and have library and partitioning facilities. The facilities of the structure should be modifiable to the experience of the searcher. There should be a minimum number of reference strategies and minimum distance between search positions.

Documentation should have stiff splines and small, thick, stiff, light pages with simple content adjustable to the experience of the searcher. It should be standardised and have a minimum number of pages and facts. Facts should be small, logically placed, distinct and have a minimum number of reference strategies. The document should have library/partition facilities.

5 New Developments

5.1 Introduction

Systems development phases are now reviewed in terms of the scanning processes, the optimal conditions and an attempt to computerise the procedures of systems development. We must collect the information into the computer from humans before we can process any data. We must prepare reports for the human so that the data can be modified and monitored. It is essential to hold information in the computer for processing, checking and modifying.

5.2 Specification

The specification phase collects input from the user against a set of rules to produce an output document. The theory shows that the input, rule set and output should be coordinated. The combination should be standardised, small, simple, specialised, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There are a minimum number of reference strategies which are modifiable according to the experience of the user and analyst.

In terms of a computerised project work bench, the system solicits the information from the user and stores it in a library which is later processed to give a report for the analyst. Most of the properties restrict the interactive language with the user and allow the dialogue to be modified according to experience of the user. The computer takes the place of the systems analyst and generally compares well in characteristics with the analyst provided we can simulate the learning aspects. The library/partitioning uses a keyword indexing method for retrieval, analysis and prompting. Modification of the specification can best be completed using a light pen and vocal input system.

5.3 Coding

Coding scans the program specification and programming language specification to produce the program. Theory demonstrates that the specification, the language rules and the program should be coordinated. The combination should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There are a minimum number of reference strategies which are modifiable to the experience of the programmer.

The project work bench reflects the actions of programmer from two approaches. The first is to follow the prompting technique outlined in the specification section. The second method performs a direct translation from the specification into the program with a prompting approach in cases where the definition is not sufficiently detailed. The impact of language definition gives properties of enhanced reliability, easy learning, running efficiency which can be built into the coding process and prompts.

5.4 Data Preparation

Data preparation scans the program and the encoders keyboard/menu pad to produce a computer readable program. The theory shows that the program and encoder actions should be coordinated. The combination should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning/programming facilities. There should be a minimum number of production strategies which should be modifiable to the experience of the operator.

In the case of the project work bench, this phase is eliminated when defining computer programs. For data collection we can use direct data entry devices, script readers, document readers, voice input, keyboard or light pen and menu which are described in other sections.

5.5 Testing

Testing requires scanning the specification for test data, the program for test points, the test manual for test methods, the test output for errors and the program for corrections. Theory shows that the specification, program, test manual and test data should be coordinated as should be the test manual, the test data, the test output and program for errors and test data, test output, program errors and program corrections. Each of the combinations should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There are a minimum number of reference strategies which are modifiable to the experience of the tester.

The project work bench can take the two approaches which were taken in the coding. The first approach is to force the user to scan the specification and provide the system variable values (possibly from a library) so that input values are used to calculate output values which were already defined (this is equivalent to a value trace and does away with the need for dumps). The system records what corrections of the library are checked as valid - equivalent to subroutine rollback or label trace. The simulator, message generator or systems exercisor are resolved by the planning analysis being applied to the computer system resources allowed for each part of the programming system. The second approach consists of proving the system correctness and showing that all ranges of input parameter are catered for. Both approaches can be useful in forcing bugs out of a programming system.

5.6 Documentation

Documentation stages scan program code, specification, previous documentation and documentation rules to produce updated documentation. Theory forces the scans to be coordinated. The combination should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning/programming facilities. There should be a minimum number of reference/production strategies which should be modifiable to the experience of the documenter.

The theory shows that the documentation should have stiff splines and small, thick, stiff, light pages with simple content adjustable to the experience of the user. It should be standardised and have a minimum number of pages and facts. Facts should be small, logically placed, distinct and have a minimum number of reference strategies. The document should have library/partition facilities.

The project work bench should have a library of different user approaches for different topics. If data is not provided for particular topic approach, then the other approaches for that topic will be listed so that the documenter can select the nearest approach which he can use as a base of information to produce the new approach from the prompts of the approach. When an approach to a topic needs to be updated, the approach is solicited so that the update can be made. When an update of a topic approach is made then a stepwise strategy for topics and approaches is made originating from where the first update was made. Updates can be made through vocal, script, document, light pen, or VDU input device.

5.7 Planning

The planning phase of systems development requires the scan of the specification, the progress/problem reports and planning rules to produce a plan. Theory results in the specification, the reports, the planning rules and the plan being coordinated. The combination should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There are a minimum number of reference strategies which are modifiable to the experience of the planner.

The project work bench scans the specification library in conjunction with the report reading peripheral and manpower data library to evaluate a Pert network and produce a plan report on a graphical output device. Budgeting information for the project and the system constraints can be evaluated by the network to ensure that those constraints are met.

5.8 Communication

We collect information informally by conversation and letter and formally by reports. Some of the information requires recording and some not. The communication process scans the organisation for a source and a destination and the information universe for the information content and transmission mode. The theoretical treatment of the subject shows that the source, destination, information content and transmission mode should be coordinated. The combination should be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There should be a minimum number of sending strategies which should be modifiable to the experience of the personnel of the organisation. Information should be as light as possible.

For the project work bench we need to use voice input and voice output to correspond to a conversation. A script reader corresponds to letter transmission. When a report is transmitted by a document reader and a hardcopy or VDU output, we have the delivery of a formal communication by messenger. If we need to find the source of information or destination of information in the organisation, then it is necessary to consult a display of the full organisation structure or of a portion of the organisation structure driven by a description of the functions or information content. When information is transferred it is logged on a central file for later analysis. The coordination of source and destination means communication software. The other properties define the properties of the language used to control the information flow. The library/partition facilities reflect the partitioning of the organisation into departments and the use of translation and edit facilities and partitioning of messages.

5.9 Personnel

The personnel section of systems development scans the organisation structure, the job definitions, organisation policy, personnel attribute rules and selection rules to give new personnel. The organisation structure, job definitions, organisation policy and personnel attribute rules scan processes must be coordinated and so must be the selection rules and applicant. The combinations must be standardised, small, specialised, simple, logically organised, have minimum ambiguity, minimum number of error cases and library/partitioning facilities. There are a minimum number of reference strategies which are modifiable to the experience of the personnel officer.

Other conclusions from the theory of previous subsections of this section are that a person should be experienced especially in the field of interest. He should be a good worker (accurate, efficient, good memory, careful, precise, fast learner) who is able to settle to work quickly and concentrate for long periods. He should use his memory rather than documentation. If he is forced to use documentation he should have supple joints, long, light fingers which allow pages to slip through them when making a reference. Motions should be kept gentle and within the middle range of movement and concentrated to the fingers only. The person should have natural dexterity, aptitude and fast recall and fast analysis. He should adopt the most effective strategy possible.

The project work bench assists personnel management by displays of the organisation structure, planning reports, job definitions and personnel records. Matching personnel to jobs is a simple scan of the organisation, job definitions and personnel records to find the nearest match. There is clearly a need for library facilities and update facilities as in previous sections.

6 Hardware System (See Figure 2)

The research has led to a hardware configuration regarded as a project utility tool similar to the programmer's work bench concept. This configuration includes:

(1) library storage for data/control library
(2) hardcopy for graphic output
(3) graphic input/output unit for amendment and checking information
(4) light pen unit for control input
(5) keyboard for character input
(6) document reader for mass data input
(7) vocal input/output unit for data transfer and amendment
(8) script reader device for collection of informal communication.

7 Software System

The work leads to a software system with the following components:

(1) device drivers for the hardware configuration described above
(2) a database system able to solicit for update data

(3) a database system able to accept data in any order
(4) a system able to analyse consistency and completeness of the database
(5) a system able to provide data of different details according to the end user
(6) a system able to analyse a subset of the database to the optimum structure and data form.

8 Database System

The database defined by the research has:

(1) a library of topics
(2) a library of user approaches per topic
(3) a library of rules per user approach
(4) a library of user experience data per user approach
(5) a set of macro facilities
(6) a set of partitioning rules for the database
(7) the rules are specialised, standardised, have minimum ambiguity, minimum complexity, minimum number of rules per component, minimum distance between sucessive application of rules, minimum number of alternative subrules, be self consistent, self checking, mutually redundant, short
(8) the library should use sets of common data, rules etc. where possible
(9) the library should be efficient to use
(10) the rules should be easy to learn and apply.

9 Further Work

So far, we have only defined the overall system requirements - these need to be refined into full specifications. The classification of topics and approaches for documentation need to be standardised along the lines of the ACM classification or Library of Congress system. We need to centralise our program systems information in a New Zealand library run by the DSIR or Computer Society such as the ACM Algol Algorithms or National Computer Centre in the U.K.

10 Conclusions

We have outlined a computerised project work bench system which is based on the results of a theoretical approach to systems development. The system is based on some hardware which is not yet fully developed and some software that is of the future. It is clear that there is a long way to go in this area but it should be developed and so bypass many of the troubles of systems development.

11 References

(1) Lomax, J.D., Documentation of Software Products, NCC, Manchester, U.K., 1977

(2) Hetzel, W.C., Program Test Methods, Prentice-Hall, Englewood Cliffs, N.J., U.S.A., 1973

(3) Naur, P. and Randell, B. (eds.), Software Engineering, Brussels, NATO Science Committee, 1969

(4) Buxton, J.N. and Randell, B. (eds.), Software Engineering Techniques, Brussels, NATO Science Committee, 1970

(5) Chandor, A., Choosing and Keeping Computer Staff, George Allen and Unwin, London, 1976

(6) Brown, A.R. and Sampson, W.A., Program Debugging, Macdonald & American Elsevier, London, 1973

(7) Bauer, F.L., Advanced Course of Software Engineering, Springer-Verlag, Berlin, 1973

(8) Brandon, D.H., Palley, A.D., and O'Reilly, A.M., Data Processing Management, MacMillan, New York, 1975

(9) Yourden, E., Techniques of Program Structure and Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1975

(10) Buckle, J.K., Managing Software Projects, Macdonald and James', London, 1977

(11) Wegner, P., Programming Languages - The First 25 Years, IEEE Trans on Computers, Vol. C25, No. 12, Dec. 1976

(12) Van Wingaarden, A., et al., Report on the Algorithmic Language Algol 68, Numer Math, Feb. 1969

(13) Cobol 1961 Revised Specifications for a Common Business Oriented Programming Language, U.S. Government Printing Office, 1961

(14) Teichoew, D. and Savayi, H., Automation of System Building, Datamation, Aug. 15, 1971

(15) Davis, C.G. and Vick, C.R., The Software Development System, Proc IEEE/ACM 2nd Int Conf Soft Eng, Oct. 1976

(16) Stay, J.F., Hipo and Integrated Program Design, IBM Systems Jnl. 13, No. 2, pp. 143-154 (1976)

(17) Boehm, B.W., Software Engineering, IEEE Trans on Computers, Vol. C25, No. 12, Dec. 1976

(18) Peters, L.J. and Tripp, L.L., Comparing Software Design Methodologies, Datamation, Nov. 1977

(19) Van Cott, H.P. and Kincade, R.G., Human Engineering Design, McGraw-Hill, 1972

(20) Ferrara, R. and Nolan, R.L., New Look at Computer Data Entry, Jnl. of Systems Management, Vol. 24, No. 2, Feb. 1973

(21) Rogers, E.M. and Rogers, R.A., Communications in Organisations, MacMillan, New York, 1976

(22) Seemuller, G., Language Aspects in Operating Systems in Language Hierarchies and Interfaces, Eds. Goos, G. and Hartmanis, Springer-Verlag, Berlin, 1976

(23) Payne, A.J., The Theory of Search Applied to Systems Development, Australian Computer Conference, Sept. 1978

(24) Ivie, E.L., The Programmer's Workbench - a Machine for Software Development, Comm. ACM, Vol. 20, No. 10, Oct. 1977

Figure 1: System Development Processes

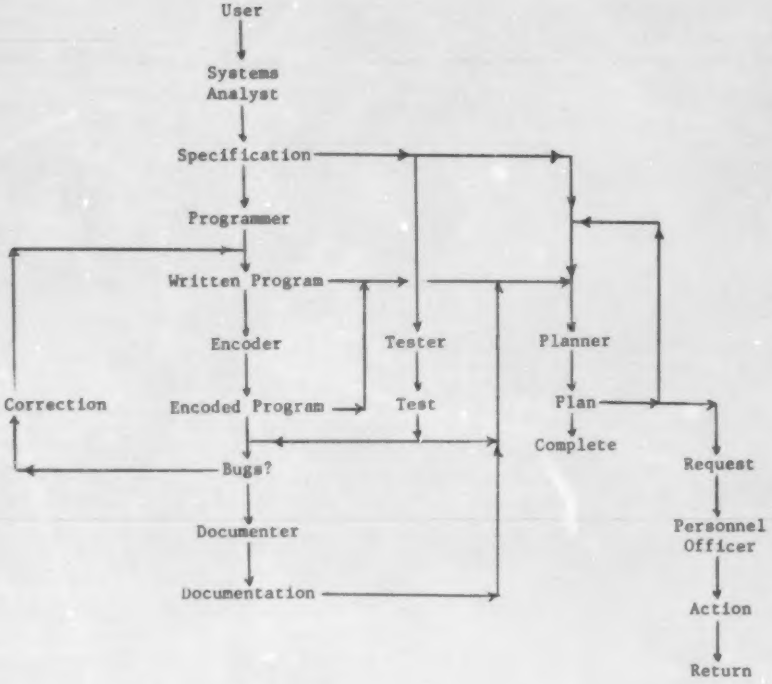
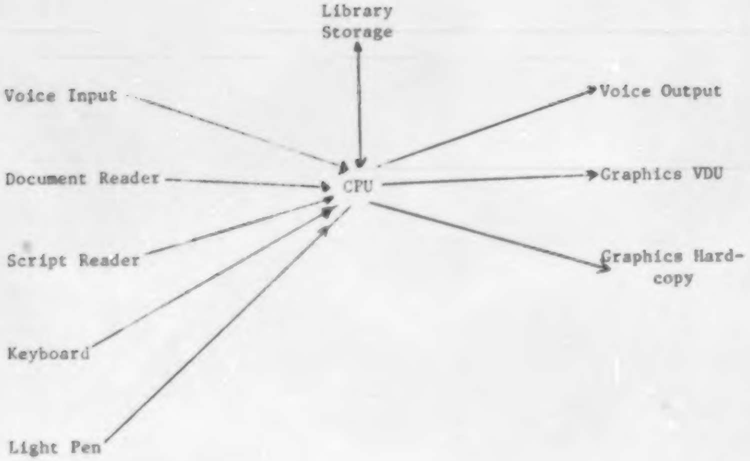

User
Systems Analyst
Specification
Programmer
Written Program
Encoder
Encoded Program
Bugs?
Correction
Documenter
Documentation
Tester
Test
Planner
Plan
Complete
Request
Personnel Officer
Action
Return

Figure 2: Hardware Configuration


Library
Storage
Voice Input
Document Reader
Script Reader
Keyboard
Light Pen
CPU
Voice Output
Graphics VDU
Graphics Hard-
copy

DEPARTMENT OF HEALTH PROJECT

AUTHOR: MR B.A. SAY
Regional Manager of EDP Services,
N.Z. Department of Health,
Auckland.

ABSTRACT:

A review of the Health Department's network system from its inception to the current status. Details include both the hardware and software currently operational, and under development.

KEY WORDS AND PHRASES:

Health project, network, DECNET, TOPS-10, transaction terminal cluster controller, transaction processing.

C.R. CATEGORIES:

2.11, 2.12, 3.34, 3.81, 3.83, 3.89, 4.35

1. INTRODUCTION

The health services of New Zealand have an annual budget in excess of N.Z. $600 million and employ 50,000 people. The Department of Health is the government agency charged with the responsibility for administering national health service programmes, e.g. medical and pharmaceutical reimbursement, public health functions, etc and acting as the executive and administrative arm of the Minister of Health. The 29 Hospital Boards, and local authorities are charged with the provision of local services ranging from the provision of hospital services, inpatient and outpatient, general and phychiatric and a range of extramural services such as home delivered meals for the elderly and infirm. Hospital Board budgets range from NZ $80 million to NZ $ 350 thousand per annum.

Since May of 1969, the Government endeavoured, through the National EDP Committee (Health) to 'co-ordinate and promote' the use of computers in the hospital service of New Zealand. Four geographic regions were established and the major Hospital Boards in each region were designated as the primary providers of EDP services for that region. Hospital Boards undertook locally initiated, but nationally approved, systems development projects which were planned for implementation on a nationwide basis.

Dissatisfaction, both in Hospital Board and in the Department of Health, led the Government to approve the employment, in mid-1974, of consultants to propose an organisation structure to achieve the objectives of the maximum utilisation of limited EDP resources by the introduction of standard systems providing operational, administrative and long and short term planning improvements. The firm of Touche Ross & Co. was selected after a tendering process involving multinational and New Zealand based firms. Touche Ross & Co., delivered its final report to the Government on 3 February 1975.

The Government agreed to adopt the recommendations considered in the Touche Ross report and made financial provisions accordingly. Touche Ross & Company were retained for two years for the implementation of the recommendations which include:-

(a) Establishment of a centrally controlled regionalised health computer service organisation as a part of the Head Office of the Department of Health.

(b) The creation of two data processing centres located in Auckland and Christchurch.

(c) The calling of proposals and subsequent negotiations of contracts for the supply of computer equipment.

(d) The initial utilisation of sub-contractors and systems and programming personnel.

2. THE HEALTH PROJECT

The consulting engagement started in July 1975 with the tendering and acquisition of two mainframe computers for the two data centres. This exercise was completed in October 1975. In parallel with this activity, work was started on the construction of data centres in Auckland and Christchurch. To speed the process, existing buildings were sought which would readily convert to operation facilities or offices to which the operational facilities could be added. This was achieved with the Christchurch Data Centre but the Auckland Data Centre was built to specification, and took a year longer to complete.

In November 1975 work was started on the three major "core" systems i.e.

- National Payroll System
- National Clinical Laboratory Computer System (N.C.L.C.S.)
- National Admissions & Discharge System

starting with the definition of requirements.

A major problem presented itself here in that network requirements should follow application system requirements but the network had to be installed to service the implementation of the application systems; a chicken and egg situation.

The three project teams submitted their data for network design in March 1976. This consisted of detailed tables of characters transmitted in both directions for hourly, daily, weekly and monthly peak traffic volumes, terminal requirements and anticipated response time needs.

3. APPROACH TO NETWORK DESIGN

Given the geographic environment and the problem of maintaing project momentum a number of premises were established. These were

- Distributed processing would be used as much as possible.
- Terminal locations dictated a concentrator approach to minimise line costs.
- A full duplex star network would be used to eliminate multidrop polling situations.
- Control should appear to be in the hands of the user.
- Transactions should be capable of being implemented by application teams.

The concept of a Transaction Terminal Cluster Controller (TTCC) was adopted to serve the dual purpose of "concentration" and terminal control. This work station would support a number of terminals, line printers and magnetic tape; perform data validation and stage transactions for sending to the main frame.

Tenders were returned in April 1975 and followed the timetable below:-

- Completion of evaluation process to short list.	July 1976
- Site visits and negotiations with short listed suppliers.	Oct 1976
- Selection of DEC to supply . terminals and concentrators . software for the TTCCs	Dec. 1976

The software requirements were developed jointly by DEC and Department of Health during November and December 1976. This approach subsequently proved a problem as it appeared that:-

- . DEC had a pre-conceived solution
- . DEC was committed to an approach on design
- . DEC had committed itself to providing a generalised package which it could subsequently market.

The software was to be developed between January 1977 and November 1977. Agreements on requirements took several weeks longer than anticipated and this together with the unexpected complexity of the project has caused the project to over-run.

The lessons to be learnt from these activities are:-

- Define your own requirements and be specific
- Ensure effective project management by the supplier
- Develop the software on site not in Marlborough USA

On balance it must be said that these problems have been larely overcome, but not without additional cost in terms of time and money. The software, when implemented, will have some rigidity inherent from using standard software modified to fit new requirements, but has the advantage of ease of software maintenance.

4. NETWORK COMPONENTS

Hardware

The hardware that has been chosen to implement the network consists of four major parts:

- . the DEC10 front end (DN87)
- . the transaction terminal cluster controller (TTCC)
- . the remote instrument cluster controller (RICC) for clinical laboratory device control
- . the terminals

The front end chosen for the DEC10 was the DN87. Two of these are connected to each DEC10 to support the eventual configuration of remote machines.

The DN87 allows for asynchronous lines for timesharing terminals as well as synchronous lines for connection of the TTCCs and RICCs. All transaction terminals will be connected to a TTCC.

The transaction terminal cluster controller is the processing computer which handles the major part of the transaction validation and control. It is a configuration based on a PDP11/34A with 65K of MOS memory and a moving head disk. A typical configuration will consist of:-

1 11/34A 64K of MOS memory
1 DMC11 synchronous interface
2 DZ11 asynchronous interfaces
1 RK05J disk (maximum of 4)
1 LP11 line printer (maximum of 4)
16 VT61/LA180 terminals
Optional TS03 Magnetic tape unit.

Laboratory instruments are controlled by the remote instrument cluster controller (RICC). This is also based on a PDP11/34A but with 48K of memory and a floppy disk rather than the RK05 disk of the TTCC. The configuration also includes the necessary analogue/Digital hardware to control the Laboratory equipment.

The terminals which were chosen to be the main transaction terminals were the VT61s. These terminals are 2000 character buffered displays with limited control to provide a protected form-filling environment with block mode data transmission.

Software

The software package being developed by DEC is based on:-
- RSX 11M
- DECNET 11M
- DECNET COMPATIBLE PORT IN DN87
- TOPS-10

and falls into four logical parts:-
. The Transaction Definition Language (TDL) compiler
. The TTCC control software and TDL interpreter
. NETSPL - a modified printer spooler to allow printer control via the message control system

. REMDEV - a DECNET package to allow file to file transfer between the 10s and 11s

Advantages

The advantages of adopting this approach are:-

- Flexibility, in that the transaction design can be easily modified.
- Ability to adapt to future growth of network, and applications by not requiring changes to the network software.
- Provides small incremental growth to the network.

However, these advantages have been at the expense of having to develop a more complex software system than would have been required by a more basic approach. Additionally there was no significant experience involving use of a transaction definition language on the DEC10 and a new language had to be specified. The lessons learned to date from the software development process are:-

- Standard software was used as far as possible in order to reduce the development time scales. The constraints imposed by this approach carried with them the built-in problems inherent in any standard software.
- The problem of developing the network software at the same time as the base applications. To be successful good communications must exist between both projects.
- Interfacing two ranges of DEC equipment (10 & 11) highlighted the incompatibilities of the communication-software.
- Communications between Marlborough (USA) and New Zealand dictate that any future development should occur in New Zealand to reduce the tremendous amount of administrative overhead associated with the long distance development effort.

WHERE ARE WE NOW?

Things are not all black. The Department and consultants are three years into the project, and although running a little late most objectives have been achieved.

The Department does have:-

- Two operational data centres
- Two installed main frames
- 35% of the network installed with the remainder ready to be installed when needed.
- Over half of the converted older systems running operationally

- Payroll running operationally on the prototype site and two others, and being implemented in five more Boards.
- Admissions and Discharge implemented in five hospitals and rapidly proving a very acceptable system.
- Laboratory system entering the live implementation phase with installation of the first hardware.
- Over 130 people recurited for the D.P. Division.
- All top management in place.
- A reduced need for consultant help.

This has been an enormous project involving nearly 400 Department of Health staff, Hospital Users, consultants and contractors, over the past two to three years. There have been problems and difficulties, but these have been resolved and overcome to the credit of all involved.

This will be the first Nationwide Health Service Computer System, and New Zealand does provide an environment in which this can be achieved. This has not been without major difficulties such as:-

- An appreciation of the length of time taken to process approvals through Government.
- The political elements of such a project.
- The difficulty of recruiting skilled staff both in New Zealand and from overseas.
- The unpreparedness of users moving from a positon of little or no D.P. experience to a full on-line environment.
- The need to build up the DLC organisation in the New Zealand environment.

To have achieved so much in three years has meant that all activities have been conducted in parallel, Communication had to be maintained at a high level to ensure problems were exposed and misconceptions eliminated.

To have conducted the project on normal sequential lines would have meant a time scale closer to 5 years, an increased cost of such a long endeavour and the lack of visible success.

However, the project is challenging and exciting and we are seeing a data processing organisation being created to meet the needs of the New Zealand Health Service. The new organisation is competent and highly professional and should provide a very firm base for launching the next series of systems projects when the current efforts have been completed.

ACKNOWLEDGEMENT:

This paper is based on a presentation prepared by Mr Say and Mr R. Hicks (Supervising Software Programmer) of the Deparment of Health, and presented by Mr Hicks to the last DECUS Conference in Australia (August 1977).

FLESHWARE AND THE COMMUNICATION PROBLEM

AUTHOR: RICHARD J. SPENCE,

Senior Lecturer,
Hawkes Bay Unit,
Department of University Extension,
Massey University.

ABSTRACT:

This paper is concerned with the frequent failure of human communication between EDP professionals and those outside the profession - the users.

The basic thesis is that communication failure is related to the attitudes and stereotypes the two groups have of each other. Accordingly, two questions are asked: How do users regard DP people? And is their stereotype of us the same as our image of ourselves?

To discover whether these differences really do reveal themselves in New Zealand enterprises, investigations were undertaken in five local organizations. The results of this enquiry are reported in the paper, together with some observations on the consequences for human communication and methods for improving it.

KEY WORDS:

programmers, analysts, EDP stereotypes, 'computniks' EDP professionals, EDP users, user's department.

C.R. CATEGORIES:

2.10, 2.12, 2.19, 2.2

"The brave new world of on-line enquiry control of business is on the other side of a growing communications gulf. The responsibility for crossing it rests on the datacrats who are conscious of its existence."

- T.B. Ward, in "The Medium is the Barrier"(1970, p.36)

Some months ago a group of us visited the four largest EDP-using companies in a provincial city. We talked with DP professionals, then with those who rely upon their service. For the users we had just one question:

"What are the main problems you face in dealing with computers?"

The reply:

"Communicating with computer professionals."

Some of the things we were told were hilarious, others sad. There was the DP manager whose view was that if users had a problem, they could hand it in at the door of his department, and his staff would hand the solution out when they had found it. Another complained, "Line staff here won't get involved in systems planning at the design stage," and five minutes later those same line staff were telling us, "The computer people don't consult us when they're designing a system for us to use." We were treated to anecdotes - some obviously apocryphal - which told us this at least: computer people have a poor opinion of users generally, and vice versa only more so.

Experience shows that in enterprises making significant use of computers for information processing, the presence of the computer often has a divisive effect on human relations. Between the groups created by the computer's presence communication is less than perfect, and critical problems arise when DP people fail to communicate with those outside the profession. Splendid diagrams like the one produced by London (1976, p.4 reproduced here as Fig. 1) look nice but only make sense if there really is effective human communication.

Communication failure is often related to the attitudes and stereotypes that people bring to the interface. Stereotypes do not by themselves cause communication failure. But it is true that line management, clerical staff and others who rely upon computer systems without being professionally involved in EDP, have built up for themselves a stereotype of the quintessential 'datacrat' (their word - a less pejorative one would be 'computnik'). Where that stereotype is unfavourable or inaccurate, communication will suffer for sure.

Can we find this stereotypic computnik? What's he like? Does he really exist? Is there even a consistent folklore about him?

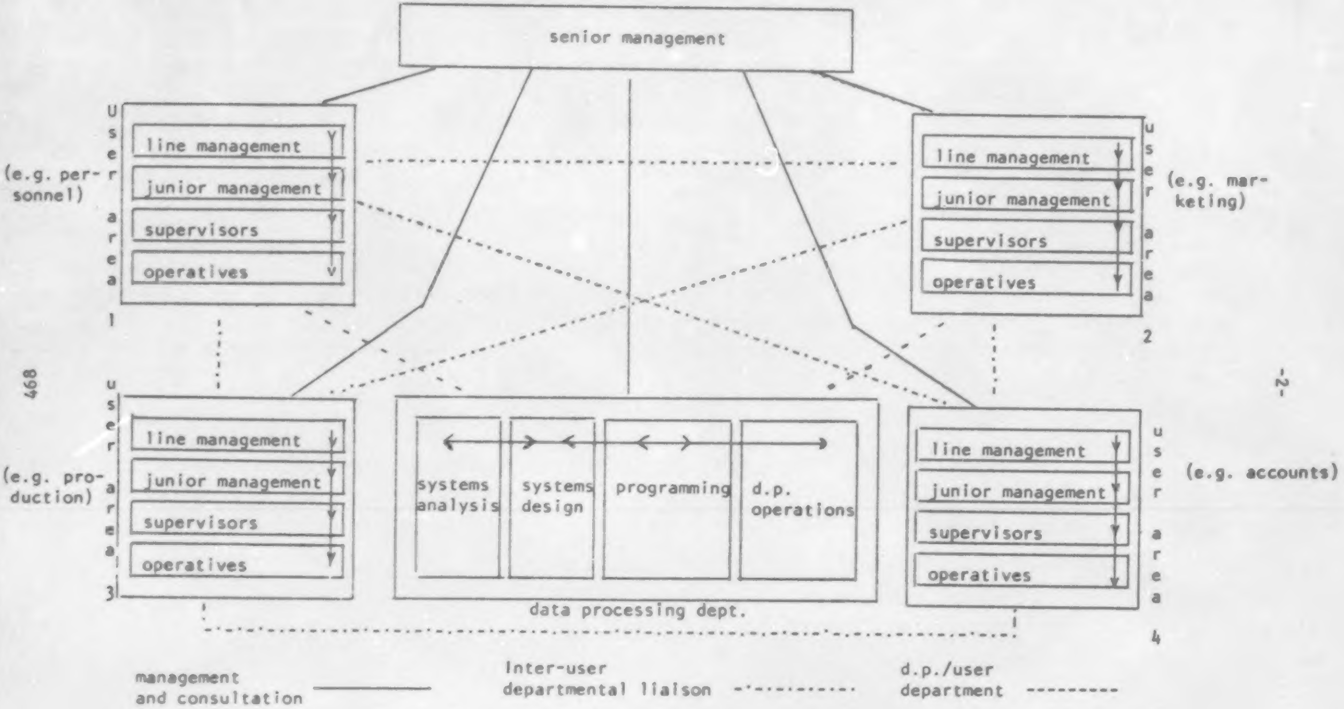

senior management
user area 1
(e.g. personnel)
line management
junior management
supervisors
operatives
user area 2
(e.g. marketing)
line management
junior management
supervisors
operatives
user area 3
(e.g. production)
line management
junior management
supervisors
operatives
systems analysis
systems design
programming
d.p. operations
data processing dept.
user area 4
(e.g. accounts)
line management
junior management
supervisors
operatives
management and consultation
inter-user departmental liaison
d.p./user department

Figure 1

What the Management Writers Say

There are many writers who tell managers how to use EDP systems to advantage. Only a few give serious consideration to the characteristics of EDP personnel (fleshware) as part of the system. I found ten, and they are listed below.

These ten writers each had something to say about the characteristics of EDP people all right, and it was remarkably consistent. The recurring theme was that computniks are young[a], arrogant[b] and highly paid[c]. They are largely ignorant of business methods[d] and more concerned with techniques than their applications[e]. What computer people don't do, according to these writers, is to identify with the company that employs them[f], to fit into existing career, pay or responsibility structures[g] or to communicate well with non-computer people[h]. They are prone, said almost every writer, to use incomprehensible jargon[i].

What the Psychologists Tell Us

Some experimental psychologists have addressed themselves to the problem of finding out the typical personality characteristics of analysts and programmers. These investigators have employed their arsenal of batteries questionnaires and statistical whizzbangs to lay bare the computnik's personality: they find him poised and self-confident[j], reflective and introverted[k] yet flexible[l]. One study provides the extra information that programmers tend to be "youthful, single males, each of whom is the oldest child in his family."[m] Programmers and analysts, say the psychologists, tend to be neither impulsive[n], nor persuasive[o] nor sociable[p]. Neither do they match the average citizen in conformity, conscientiousness, dependability and alertness to moral and ethical values[q].

---

[a] Ward (1973) pp xiv, 1; Contos & Spence (1978); Stewart (1971) p196

[b] Donaldson et al (1976) p138; Ward (1973) p xiv; Contos & Spence (1978)

[c] Donaldson et al (1976) p138; Jenkins & Serman (1977) p58

[d] Ward (1973) p1; Donaldson et al (1976) p 138; Contos & Spence (1978); Unknown, p 219

[e] Ward (1973) p92; Donaldson et al (1976) p45; Rothman & Mosmann (1972) p42

[f] Ward (1973) p92; Donaldson et al (1976) p45; Contos & Spence (1978) McFarlan et al (1973) p340; Unknown p203

[g] McFarlan et al (1973) p340; Jenkins & Sherman (1977) p15; Stewart (1971) p196; Donaldson et al (1976) p138; Unknown p203, 217

[h] Sanders (1973) p24; Donaldson et al (1976) p45

[i] Ward (1973) p1, 50; Sanders (1973) p24,76; Jenkins & Sherman (1977) p15; Donaldson et al (1976) p45; Stewart (1971) p196; Contos & Spence (1978) Mumford & Banks (1967) p216

[j] Morris & Wise (1972)

[k] Alspaugh (1972); Barnes (1974)

[l] Morris & Wise (1972)

It's a useful and suggestive exercise to write down the list of attributes that the management writers claim for EDP personnel, alongside what the psychologists have found. The match is better than fifty percent, and even where they do not match exactly, there is no conflict: it's just that the two groups have been dealing with different issues. The literature does reveal some kind of stereotype, then, and it does seem to have some basis in fact.

What Computniks Try to Be Like

Having considered the generally unfavourable view of EDP people offered by the management writers, and the professionally noncommittal stance of the psychologists, it is worth a slight detour to see what the profession thinks of itself - or rather, what it hopes to be.

Let's start with programmers. Most would agree with Gerald Weinberg (1971, p 150) that the key qualities are the ability to combine humility with assertiveness, to be adaptable to rapid change, and to have "a slight compulsion to keep one's papers in order."

Analysts, too, need flexibility combined with an orderly mind, according to Daniels & Yeates (1971), p1. But in addition they need to be intelligent, perceptive and persistent, and must have "strength of character" (whatever that may mean - can it be a polite word for assertiveness?). Furthermore, analysts must be experts at human communication:

> "The systems analyst must be able to express his thoughts, ideas and proposals clearly, both orally and in writing. In addition, he must be a relaxed listener, and ... an accurate conversationalist. He must have more than average social skill ..."

And for EDP managers, says Nolan (1973, p145),

---

[m] Barnes (1974)
[n] Alspaugh (1972)
[o] Morris & Parkinson (1971)
[p] Alspaugh, op. cit.
[q] Morris & Wise, op. cit.

> "A good understanding of the trends of computer technology is still essential ... but it is only one ingredient ... the EDP manager must have a good understanding of his company, its capacity for change, and its competitive environment. He must also deftly execute a continuing mission to educate functional managers about how computers can be used in the business. To solve these problems, he will have to revise his own department's organization in some ways, and this is a further challenge."

Summary of the Review of Literature

It is apparent from this review that the literature reveals very little agreement between the computnik's stereotype of himself and the manager's view of him. The one point where they might agree is on the DP man's assertiveness ('arrogance', if you prefer), but elsewhere they conflict directly. Psychological evidence supports the one stereotype in some respects, elsewhere it supports the other.

The New Zealand Interviews

In an effort to resolve some of the issues that the literature raises, the author undertook a series of twelve group interviews with 100 clerical and administrative personnel in three large New Zealand organizations.* Not only would these interviews contribute to an understanding of the situation in this country; it could also yield further information on a related and popular theory.

This theory, which enjoys widespread currency in managerial circles, says that any person's attitude to EDP in general will be greatly altered if he "gets interactive" with some system(s). Users of interactive facilities, however technically naive, are said to become more positive in their attitude toward computer systems - and computer people. Empathy between user and computer professionals is seen as a side-effect of the user's reciprocal relationship with the system.

A reciprocal relationship was defined for the purpose of this study to be one in which the same person receives output from input that he himself has generated. This definition includes much work done in batch as well as conversational mode; it excludes dataprep (when operators don't see their output) and equally excludes managers who get output from other people's input. Two-way interaction is the defining issue, and it was decided that the person best qualified to judge whether such interaction existed was the respondent himself. Accordingly, Question 1 became an integral part of the protocol used in the interviews.

---

* These organizations are so well-known that any description, however vague, would instantly identify them.

The complete protocol covered:

(1) the respondent, his place in the organization and his experience with EDP

(2) his perceptions of:

  (a) the behavioural characteristics of EDP people in the organization

  (b) their attitudes

  (c) their personal traits

This protocol (detailed in Appendix I) deals with every issue discovered in the literature search. To prevent the researcher's own (perhaps unconscious) biases from influencing the respondents' replies, Roget's Thesaurus was used as the source of synonyms and atonyms used in the questions. A further safeguard would have been to hire an independent interviewer, but this was financially infeasible.

The Problem of Classification

It was anticipated that there would be some difficulty in separating EDP professionals from EDP users in some organizations. This proved indeed to be the case; only one of the organizations studied had actually implemented the classical model of compartmentalised functions implied by Fig. 1. (It is interesting that that was the one whose communication problem appeared most acute.) Question 2 of the interview protocol was used in an attempt to discover which respondents were "insiders" and which "outsiders"; it succeeded only in part. It was effective in identifying people who were definitely outsiders, but it failed to locate insiders with precision. This was evident in certain interviews with people who despite having responded "over 6 months" to Question 2, nevertheless persistently spoke of EDP professionals as "them", implying that possession of a certain job title did not necessarily make one a "computer person."

Insofar as the questionnaire distinguished between different people who definitely did fall into the user category, but failed to identify

EDP professionals unequivocally, the useful data emerging from the interviews relate to the effects upon a user's attitudes of his getting interactive with some system(s). To obtain data on whether the computnik's stereotype of himself tallies with the users' view of him, will require some revision of the research technique.

What the Interview Revealed

There was general consensus among all users interviewed that computniks are dedicated and ambitious career professionals. They were also unanimous that the use of specialized jargon is characteristic. Furthermore, they believed that computer prople are generally male or at any rate mannish.

These three points aside - and they were predictable - different groups of users offered different views on specific issues. Among those who had never used an EDP system in a reciprocal way, there was consensus on 16 further points. By contrast, those who often used EDP systems reciprocally, agreed among themselves on only 6 further points.

The users who had not experienced a reciprocal relationship with a computer system agreed that EDP people work fantastic hours, ignore the imponderables that form part of management decision-making, tend to have higher education and high pay. They saw computniks as being more concerned with computer techniques than with what the system can do for the organization, convinced of their tremendous contribution to the organization's effectiveness, and in need of mutual stimulation. Furthermore, in their view computer professionals are logical and intelligent yet unoriginal; arrogant and intimidating yet congenial and sociable; and excitable as well.

Those who had used EDP systems in reciprocal fashion did not disagree with what the others had said, but their consensuses were fewer in number and concerned other issues. These people agreed that computniks are poor communicators, ignorant of business methods, careless about user departments' schedules and budgets and neglectful of the clerical systems that surround the computer. They say EDP professionals as being methodical people who have little identification with the organization that employs them.

In addition to the matters mentioned in the interview protocol, various groups volunteered opinions on other related issues. They are not reported here, but have been incorporated in a revised protocol (Appendix II).

Commentary

Even from this preliminary study, it is established that there are differences in attitude between those computer users who had used systems in reciprocal fashion, and those who hadn't. It is particulary interesting to note the stronger tendency of the latter group to form a more detailed stereotype of the essential computnik.

An explanation for this effect might point to the general tendency for the least knowledgeable people in any field to be the ones with the most definite opinions. It might also suggest that those individuals who have been humbled by the sight of their own garbage coming back at them are less inclined to generalize about those people who do it for a living.

Whatever the explanation, it does appear that providing users with interactive facilities (eg on-line terminals) may radically alter their attitudes toward EDP personnel. This matter warrants further, more refined investigation. Larger samples will permit statistical analysis of the data, and the revised interview protocol will improve its quality.

In the meantime, the improved protocol provides a tested method of discovering important features at the man/man interface as it exists between EDP professionals and users in any given organization. It will therefore be immediately useful to companies that care about fleshware and the communication problem.

APPENDIX I: Interview Protocol Used in the Study

Question One:

Do you know, or have you at any time, used a computer system in a reciprocal way? That is, have you provided input to the computer and made use of the information that came out as a result?

☐ Often ☐ Very seldom ☐ Never ☐ Uncertain

Question Two:

Have you ever been employed in one of these occupations?

Keypunch Operator
Minicomputer Operator
Computer Operator
Input Equipment Operator
Data Control Clerk
Data Processing Manager
Systems Development Manager
Computer Engineer
Customer Engineer
Computer Marketing Representative
Systems Analyst or Designer
Trainee or Assistant Programmer
Programmer/Analyst
Programmer (includes Senior Programmer, Applications Programmer, etc.)
Data Entry Operator
Installation Supervisor

If yes, please put a ring around the ones that apply to you.

If you have had a job that is very similar to those listed above (perhaps the list is incomplete) please say what it is (or was):

____

Please say how many months altogether you have been employed in the jobs you mention (so far):

Total ________________months

WHAT DO YOU NOTICE ABOUT EDP PEOPLE IN YOUR ORGANIZATION AND ELSEWHERE?

business methods : knowledgeable/ignorant

the hours they work : conventional/fantastic

real knowledge of computer technology : superficial/considerable

about user departments schedules and budgets : concerned/careless

the clerical systems that surround the computer : care for them/neglect

the imponderables that form part of management decision-making : take notice of them/ignore them

higher education : more likely/less likely than average to have it

pay : high/low

career, pay and responsibility structures : do/don't fit into them

weight in the organization : inadequate/excessive

communication with non-EDP people : good/poor

specialized jargon : use it/ avoid it

WHAT ABOUT THEIR ATTITUDES?

more interested in what the computer system can achieve for the organization than with the methods used to get there	more concerned with computer techniques than with what the system can do for the organization
identify strongly with the organization that employs them	little identification with the organization that employs them
convinced of their tremendous contribution to the organization's effectiveness	don't believe they have much to contribute to business efficiency
are dedicated ambitious career professionals	don't worry much about getting on in their careers
are firmly in touch with everyday reality	live in a rarified atmosphere
need mutual stimulation	are self-contained and self-stimulating
working class	middle class

COMPUTNIKS TEND TO BE . . .

(60) orderly, methodical, systematic
(61) disorganized, slovenly, chaotic

(83) conventional, conformist, regular
(84) unorthodox, informal, eccentric

(130) young, youthful
(131) mature, senior, middle-aged

(144) conservative, unchanging, stable
(143) adaptable, ready to change, flexible

(224) inward-looking, introverted, "deep"
(223) extroverted, outward-looking, superficial

(372) masculine, virile, mannish
(373) womanly, feminine, effeminate

(475) logical, analytical, rational
(476) intuitive, plausible, unreasoning

(498) intelligent, brainy, bright
(499) dull, obtuse, stupid

(516) easy to understand, explicit, comprehensible
(517) baffling, obscure, unintelligible

(513) creative, inventive, ingenious
(non-513) unoriginal, unimaginative

(612) persuasive, encouraging
(613) daunting, intimidating, dissuasive

(709) quarrelsome, cantankerous, provocative
(710) congenial, harmonious, amicable

(822) excitable, enthusiastic, impulsive
(823) cold-blooded, imperturbable, dispassionate

(819) perceptive, responsive, sensitive
(820) thick-skinned, unfeeling, callous

(878) arrogant, high-handed, overbearing
(879) servile, cringing, slavish

(883) retiring, solitary, secluded
(882) sociable, free-and-easy, gregarious

(897) sympathetic, considerate, kindly
(898) caustic, churlish, ruthless

APPENDIX II: Revised Interview Protocol

Are you accustomed to providing input to some computer system? . . . . Yes / No

Do you frequently use computer output? . . . . Yes / No

If you answered yes to both the above questions, do you put information into the computer and make use of the output that results from your own input? i.e. do you use the computer in a reciprocal way? (Most people who use interactive terminals for reservations or customer enquiries would say yes: most key punch operators would say no, because they don't normally use the output that results from the input they prepare. In the same way, people who use reports that come from other people's inputs would say no as well.) . . . . Yes / No

WHAT DO YOU NOTICE ABOUT EDP PEOPLE IN YOUR ORGANIZATION AND ELSEWHERE?

business methods and terminology : knowledgeable/ignorant

the hours they work : conventional/fantastic

their real knowledge of computer technology : they understate it/overstate it

about user department's schedules : concerned/careless

about user department's budgets : concerned/careless

the clerical systems that surround the computer : care for them/neglect them

the imponderables that form part of management decision-making . take notice of them/ignore them

higher education : more likely/less likely than average to have it

total income (not just salary in this organisation) : high/low

career, pay and responsibility structures : do/don't fit into them

establish organisational structures : use them/short circuit them

weight in the organisation : inadequate/excessive

communication with non-EDP people : good/poor specialized jargon : use it/avoid it

other people understanding EDP methods : resist/encourage it

priorities, work vs. personal life : put work first/personal life first

acceptance of principle that management controls the organisation, not the EDP department : ready/reluctant

COMPUTNIKS TEND TO BE . . .

(60)	methodical, systematic	(61)	disorganized, chaotic
(83)	conventional, conformist	(84)	unorthodox, eccentric
(130)	young, youthful	(131)	mature, senior
(144)	conservative, unchanging	(143)	adaptable, flexible
(224)	inward-looking, "deep"	(223)	extroverted, outward-looking
(475)	logical, analytical	(476)	intuitive, good guessers
(498)	intelligent, brainy	(499)	dull, obtuse
(516)	easy to understand, comprehensible	(517)	baffling, obscure
(513)	creative, inventive	(non-513)	unoriginal, unimaginative
(612)	encouraging, persuasive	(613)	daunting, intimidating
(709)	quarrelsome, provocative	(710)	congenial, amicable
(822)	excitable, impulsive	(823)	cold-blooded, dispassionate
(819)	responsive, sensitive	(820)	thick-skinned, unfeeling
(857)	tense, fidgety	(856)	relaxed, nonchalant
(878)	arrogant, high-handed	(879)	servile, cringing
(882)	gregarious, sociable	(883)	retiring, solitary
(897)	considerate, kindly	(898)	churlish, ruthless

WHAT ABOUT THEIR ATTITUDES?

more interested in what the computer system can achieve for the organisation that with the methods used to get there	more concerned with computer techniques than with what the system can do for the organisation
identify strongly with the organisation that employs them	little identification with the organisation that employs them
identify strongly with the particular make of equipment they work with (Burroughs, IBM, Digital, etc.)	no particular loyalty to any make of equipment
convinced of their tremendous contribution to the organisation's effectiveness	don't believe they have much to contribute to business efficiency
are ambitious career professionals	don't worry much about getting on in their careers
welcome tasks that put them firmly in touch with everyday reality	try to live in their own private, rarified atmosphere
need mutual stimulation	are self-contained and self-stimulating
are highly involved with the work they are doing	do not care much about their work
prefer simple solutions rather than complex ones	are complicators, not simplifiers
suffer fools gladly	intolerant of other people's weaknesses

BIBLIOGRAPHY

Works Cited

Alspaugh, C. A.: "Identification of some components of Computer Programming Aptitude." Journal for Research in Mathematics Education (March 1972) pp89-98

Barnes, P. H.: "A Study of Personality Characteristics of Selected Computer Programmers and Computer Programmer Trainees" Unpublished Ed.D. Thesis, Auburn University, 1974

Contos, K. & Spence, R.: "Dethroning the Deus Ex Machina." Data Processing in New Zealand (January 1978)

Daniels, A & Yeates, D. A.(eds): Basic Training in Systems Analysis (2ed) (London 1971)

Donaldson, Stevens & Becket: Computer by the Tail (London, 1976)

Jenkins, C & Sherman, B: Computers and the Unions (London 1977)

London, K. R.: The People Side of Systems (London 1976)

McFarlan, F. W. Nolan, R. L. & Norton, D.P.: Information Systems Administration (New York 1973)

Morris, J. L. & Parkinson, M.: "Vocational Interests of Data Processing Personnel." Australian Psychologist 6 (1971) no.6, pp.19-25

Morris, J. L. & Wise, J. J.: "Personality Characteristics of Data Processing Personnel" Australian Psychologist 7 (1972) no.3, pp173-174

Mumford, E. & Banks, O.: The Computer and the Clerk (London 1967)

Nolan, R. L.: "Plight of the EDP Manager." Harvard Business Review (May-June 1973) pp. 143-152

Roget: Thesaurus of English Words and Phrases (London 1976)

Rothman, S. & Mosman, C.: Computers and Society (Chicago 1972)

Sanders, N.: The Corporate Computer (London 1973)

Stewart, R.: How Computers Affect Management (London 1971)

Ward, T.B.: "The Medium is the Barrier." Management Decision (Autumn 1970) pp31-36

Ward, Tom: Computer Organisation, Personnel and Control (London 1973)

Weinberg, G. M.: The Psychology of Computer Programming (New York 1971)

Unknown (American) Author: The Management of Information Systems

BUSINESS FORMS & BUSINESS FORMS MANAGEMENT PROGRAMS

AUTHOR

Jack Stubbs,
Marketing Manager,
Moore Business Forms Concept,
Moore Business Forms Inc.,
Glenview, Illinois, 60025, USA

SUMMARY

Modern costs of record keeping demand a reduction of paper costs without adversely affecting systems efficiency and control.

A total forms management system can provide systems analysis, forms design, acquisition, inventory management and distribution.

KEY WORDS & PHRASES

Information Processing, Forms Control, Forms & System Design, Forms Supply Management, Key Operations.

C.R. CATEGORIES

2.11, 2.12, 8.9

BUSINESS FORMS AND BUSINESS FORMS MANAGEMENT PROGRAMS

The Blind Men and the Elephant - a Hindu Fable

It was six men of Hindustan,
To learning much inclined,
Who went to see the Elephant,
(though all of them were blind),
That each by observation,
Might satisfy his mind ....

When the blind men of Hindustan went to see the Elephant, each formed an image of the beast in his own mind, based on which physical part he touched. One described the elephant as a snake, another as a tree trunk, a third as a fan, and so on.

LIMITED VIEW

Like those six blind men, many data processing managers take a limited view of business forms and the part they play in the information processing business. In the eyes of many, business forms are a commodity item, and as long as they perform satisfactorily on the high speed printer, console typewriter or what-have-you, the attention given them is minimal. Business forms deserve better than that. They are the visible, tangible link between the data processing department and its user/customers. It is enlightened self-interest to ensure that forms accurately communicate the quality and substance of the information system that produces them. How can this desirable goal be achieved in an orderly, organized way? What new developments in the business forms industry can add to the effectiveness of forms from the viewpoint of user/customers? How can the DP manager keep abreast of these developments?

These questions cannot be answered in real depth in a brief article. Hopefully, a comprehensive overview of a rather large subject can be developed. We may also conclude that there is more to the elephant than we want to believe.

FORMS MANAGEMENT PROGRAMS:

Let's begin with order and organization - a business forms management program. An adequate program of business forms management encompasses three areas:

1. Forms Control
2. Forms Design
3. Inventory Management.

Forms control, centering in a file of all forms in sequence according to their function, is the heart of the forms management program. Each file contains a form sample, specification sheet and flow chart or description, telling how the form is used. As new forms are developed, or existing forms come up for reprint, they are checked against the forms control file. New forms, if they are proven to be necessary, are designed to harmonize with company graphic standards and with other related forms. Reprint orders are checked for copy, specification or construction changes, and considered for combinations with similar forms to achieve production cost savings.

Forms design includes much more than a consideration of printer spacing limitations and manifolding characteristics. A business form is a vehicle to record and communicate instructions and information. The design of business forms for computers and their user/customers is an art and a science that still contains much unexplored territory. Obviously, the forms need to work on the computer. That is the first requirement. They also need to be easily recognized for what they are - invoice, statement etc., who they are addressed to, and what action the recipient should take. These and many other criteria make forms design a field with seemingly inexhaustible opportunity for creative effort.

Inventory management is essentially a matter of ensuring that forms inventories are greater than dangerously low and less than excessively high, according to numerical values given to those categories. Sounds simple, but it fans out into such considerations as purchasing efficiency, capital investment in inventories, cash flow, utilization of storage space, stationery stores management, stock status reporting, and so forth. Faced with rising costs of money, space and personnel, many users are turning to forms suppliers to share with them the responsibility of the forms management function, and to draw on their expertise and resources to fill this need.

SHARED MANAGEMENT:

In the shared responsibility concept of business forms management, customer and supplier are partners in a mutually beneficial relationship. From the customer point of view, these benefits include relief in a number of forms-related areas...

- Forms control
- Forms design
- Purchasing and requisitioning economies.
- Lower average forms cost through forms standardization, longer runs, production economies.
- Released capital - forms are invoiced as they are shipped from inventory.
- Assistance with inventory management.

We have addressed the first question, the question of forms design and information communications, by seeing it as an integral part of a business forms management program.

Whether accomplished in-house or with the assistance of an interested and capable supplier, an effective program of business forms management is one of the keys to efficiency in data processing and includes efficient business forms design.

The premise -

Business forms and business forms management programs are an important subset of the information processing function of any enterprise. By definition, forms management includes rationalising, designing, classifying, controlling, and managing the supply of all forms used in an organisation.

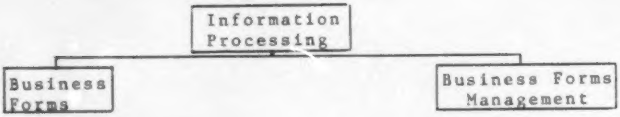
Back of the concern for an effective forms management program is a recognition of the importance of business forms in society. A business form is a vehicle to record and communicate instructions and information. Whether a form is a good communication or a bad one depends in large measure on the care, forethought and expertise of the form designer. Under the heading of forms design, we will look at some of the factors to be considered in forms design, especially computer input and output forms.

In summary, this presentation offers guidelines for instituting and maintaining a program of business forms management under three headings; forms control, forms and systems design, and forms supply management.

The emphasis throughout is on the concept of forms management as a business function. Understanding the concept comes first: application of the principles to suit particular circumstances naturally follows.

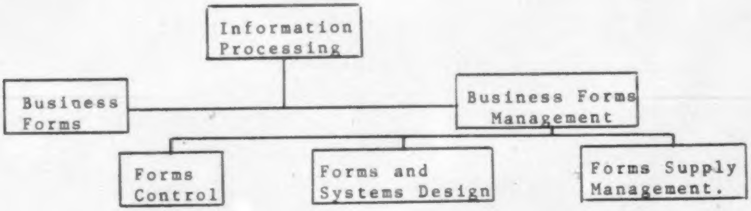
THE CONCEPT:

Information processing is a function of every business, regardless of the size of the business. Business forms and business forms management support information processing.


Information Processing
Business Forms
Business Forms Management

Business forms management has three parts....

1. Forms Control, classifying and keeping track of all forms.
2. Forms and System Design, designing new forms and form systems and improving existing forms and systems.
3. Forms supply management, maintaining forms supply.


Information Processing
Business Forms
Business Forms Management
Forms Control
Forms and Systems Design
Forms Supply Management.

Whether or not it is so identified, business forms management exists as a function wherever business forms are used. It may not always be achieved systematically, but one way or another, forms are kept track of, forms and systems design work is done, and supplies of forms are maintained.

Each of the three parts of business forms management has a number of elements.

A. FORMS CONTROL - KEEPING TRACK OF ALL FORMS

Forms File:

The forms file is the control point of the business forms management program. It contains form samples, form specification sheets, and simple flow charts to show how each form is used. It is important that the sequence of the file relates in a practical way to the way the business operates.

For instance, if a form numbering system is already well established, it may be best to set up the file in form number sequence. Otherwise, some suitable sequence will have to be established. As one alternative, the nine key operations of business:

1. Purchasing,
2. Receiving,
3. Stock-keeping,
4. Production,
5. Selling,
6. Delivery,
7. Billing,
8. Collecting, and
9. Disbursing,

may serve as a useful method of classifying forms and setting up the forms file.

Forms Catalogue:

The forms catalogue provides a convenient listing of the contents of the forms file for easy reference.

Procedures Manual Interface:

When forms are designed or redesigned, those changes must be incorporated into business procedures manuals and vice versa. A cross reference between the procedures manual and the forms file helps keep all material current.

Pre-Order Clearance:

The pre-order clearance function is a check of printing requisitions against the forms file to avoid reproducing obsolete forms and to incorporate changes. Production economies can be co-ordinated each time new and repeat orders are placed by grouping and combining the production of similar forms.

B. FORMS AND SYSTEMS DESIGN - DESIGNING NEW FORMS AND SYSTEMS, AND IMPROVING EXISTING FORMS AND SYSTEMS.

Forms Design:

Forms design is creating functional, attractive forms and improving existing forms.

Systems Design:

Systems design is problem-solving - gathering facts, analyzing, considerating alternatives, and deciding on a solution. System changes normally affect the business forms used in the system.

Forms and Systems Workshops:

These bring together systems and user department personnel to increase their knowledge of forms and systems design. Using actual situations in the subject business, these workshops can be helpful in solving problems as well as directing attention to the importance of the business forms management program.

Company Image:

Business forms are an integral part of a company's total image. Where a clearly defined image does not already exist, a forms design program can provide useful leadership in creating one.

C. FORMS SUPPLY MANAGEMENT - CONTROLLING FORMS SUPPLY:

Procurement:

Procurement describes the steps involved in selecting a vendor, purchasing, receiving and paying for the forms.

Production:

Supply management requires an understanding of the production capability of the forms manufacturing industry, including available products, product features, sources of supply etc. With this information, the buyer can select the product that best fits the user's need.

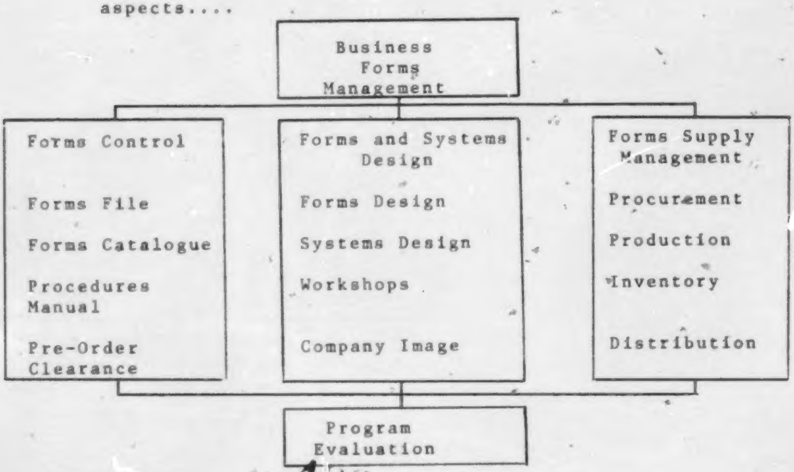
Inventory:

Inventory management is ensuring that business forms inventories are neither too high nor too low.

Distribution:

Distribution is supplying business forms to using departments from a central storage area.

We have created a model of business forms management as a business function, and defined some of its aspects....


Business Forms Management
Forms Control
Forms File
Forms Catalogue
Procedures Manual
Pre-Order Clearance
Forms and Systems Design
Forms Design
Systems Design
Workshops
Company Image
Forms Supply Management
Procurement
Production
Inventory
Distribution
Program Evaluation

IMPLEMENTING A BUSINESS FORMS MANAGEMENT PROGRAM:

In the last section, we developed a conceptual model of a business forms management program and its component parts. This section contains some suggestions for instituting and maintaining a program of business forms management under three headings:

Forms Control,
Forms and Systems Design,
and
Forms Supply Management.

The starting point for the implementation of a forms management program is the current practice, whatever is being done now. It is not a matter of trying to force-fit a theoretical model into an existing situation. The first step is an understanding of the forms management concept, then applying the principles to fit the needs of a particular situation.

With the foregoing in mind, here are some suggestions for instituting a forms management program:

FORMS CONTROL:

Advise all affected personnel that a forms management program is being instituted and outline its scope, who is responsible for the program, etc.

Collect three samples of each form in present use, and a brief explanation of how each form is used.

If no suitable form numbering system exists, establish one. The form file will be maintained in this sequence, using the form number as the file index number.

Here, for an example, is a five position form numbering system....

DDKNN,	where
DD	designates the using department
K	shows the key operation, and
NN	is a numerical identification.

Now classify the collected form samples according to the form numbering system, and assign a unique form number to each form.

Set up the form file, and add to each file a form specification sheet and a flow chart to show how the form is used.

Prepare a catalogue listing of all forms in the file, in form number sequence. This will have a number of uses, but it is first of all a control listing of all the forms in the file.

File new forms when they are designed, and changes to existing forms, in the forms file. Flag files for forms that become obsolete. Check printing requisitions against the file for any changes since the last order. Using cross reference notes, check for ordering combinations to achieve production economies in printing.

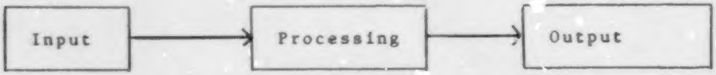
FORMS AND SYSTEMS DESIGN:

People everywhere complain about paper - the number and complexity of the forms we have to fill out, the sheet mass of paper that seems to be necessary to feed our information-hungry society.

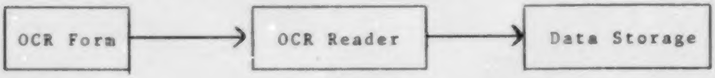
Why this complaint? The communication of information is obviously necessary in the world in which we live. And in most situations, despite the promise of new developments, paper continues to provide the least expensive, most convenient means of recording and communicating instructions and information. Is it possible that some really dramatic improvements can be made in fusiness forms to make them easier to understand and use?

No douht the consistent application of a few simple design rules would help. But beyond that, on a grander scale, we may discover a challegning new frontier for the creative mind, an opportunity to make a truly significant contribution to the lives of people in the world around.

Consider this processing diagram:

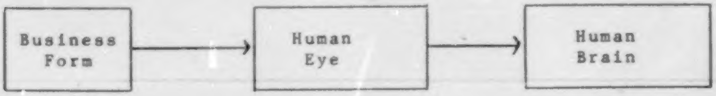


In OCR terminology, this diagram could read:



The unforgiving characteristics of the OCR Reader have obliged us to be fanatically specific in the design and production of OCR forms as to size, colour, placement of data, and so on.

Now we consider the same diagram in human terms:



No input device designed by man is so forgiving, so capable of compensating for poor data presentation, as the human eye and brain. But although they can accept poor input, humans can also complain, and they do so.

The job of the forms and system designer, then, is much more than designing acceptable forms. The complete job includes taking into account the emotional reaction of those who receive and use the forms. Forms design provides an excellent opportunity for creative work, with measurable benefits in customer and public acceptance, employee morale, accuracy and business efficiency.

We have not included a complete course in form design in this presentation, but the slides we now present depict some of these considerations.

FORMS SUPPLY MANAGEMENT:

The cost of business forms includes considerably more than the amount of the supplier's invoice:

1. Procurement cost,
2. Forms Cost,
3. Cost of capital employed,
4. Inventory storage,
5. Inventory control,
6. Distribution to using areas,
7. Cost of using the form,
8. Records storage cost.

A forms management program is a cost control program, concerned with each element of the cost of business forms. The paper and ink itself, the "forms cost", represents between five and fifteen percent of the real cost of forms. The largest element, the cost of using the form, is addressed through the forms and systems design aspect of the forms management program. That is where the greatest savings may be made.

Savings can be found in other areas too. Consider procurement cost, for instance. The procurement cycle has a number of steps:

1. Inventory check,
2. Requisition,
3. Approval,
4. Request quotations,
5. Sales intervies,
6. Purchase order preparation,
7. Expediting,
8. Receiving,
9. Accounts Payable.

These nine steps and the cost of performing them recur every time an order for printing is placed.

One of the objectives of a comprehensive forms management program is reducing the number of procurement cycles, with a corresponding reduction in clerical cost.

This may be done by combining like forms into group purchases, reducing forms cost, as well as procurement cost.

Another alternative is to have using departments draw from the supplier-held inventory by requisition. If the forms are invoiced at time of shipment, no company capital is tied up in forms inventory and forms storage space is released for other uses.

In addition, the supplier maintaining the inventory would be responsible for inventory status reports and related inventory management information.

We have explored some of the factors involved in procurement and considered some potential savings. There are possibilities for savings in other areas also, possibilities that would come to light in the normal course of events when a forms management program is undertaken.

PROGRAM EVALUATION:

Evaluating the benefits and cost effectiveness of the forms management program is essential to prove the program's value.

Here is a checklist of some of the factors to consider in evaluating a forms management program. A diary documenting events related to the program is useful in preparing the evaluation.

GENERAL OVERVIEW OF THE FORMS MANAGEMENT PROGRAM:

1. In broad terms, is the program fulfilling its objectives in forms control, forms and systems design, and forms supply management?

2. Does the program operate smoothly and efficiently?
3. Has it reduced the cost of doing business?
4. Can you justify the cost of the program?

FORMS CONTROL:

(Forms File - Catalogue - Procedures Manual - Pre-Order Clearance).

1. Are there improvements in these elements since the last review? What are they?
2. Report on the status of the forms control program.

FORMS AND SYSTEMS DESIGN:

(Forms Design - Systems Design - Forms and Systems Workshops - Company Image).

1. List new forms and revisions, with before-and-after samples. Standardization of forms, showing savings, would be included here. Arrange the examples in forms file number sequence.
2. Include examples of systems changes with flow charts of the systems involved. Are systems generally in good shape?
3. Did we conduct any workshops? How many, when, who attended, what did we do?
4. What have we done to enhance the company's image through forms design? This applies to inside as well as outside forms.

FORMS SUPPLY MANAGEMENT:

(Procurement - Production - Inventory - Distribution).

1. Is the forms procurement cycle operating efficiently?
2. How is the communications between us and the forms industry?
3. Is the inventory control system working?
4. Are using departments kept supplied on time and with a minimum of fuss?

OTHER POINTS TO CONSIDER:

1. Review progress in a general way compared to previous objectives.
2. Outline plans for the future.

The program evaluation report should be presented and discussed with senior management on a regular basis.

BENEFITS AND CONCLUSIONS:

We have developed a conceptual model of a business forms management program and offered some suggestions and comments for implementing and maintaining a forms management program under three headings:

- * Forms Control,
- * Forms and Systems Design,
- * Forms Supply Management.

The benefits to a company embarking on such a program can be summarized as follows.....

1. Control of all forms.
2. Forms and systems design efficiency.
3. Company image enhancement.
4. Improved employee morale.
5. Reduced forms procurement cost.
6. Lower average forms cost through longer runs, forms standardization, grouping of orders.
7. Released storage space if the supplier maintains the forms inventory.
8. Released capital - forms are paid for when shipped from supplier's warehouse.
9. Inventory status reports provided by the supplier.

It is perhaps worth repeating here somethin that was included earlier in the presentation:

> "The starting point for the implementation of a forms management program is the current practice, whatever is being done now. It is not a matter of trying to force-fit a theoretical model into an existing situation. The first step is undertsanding the forms management concept, then applying the principles to fit the needs of a particular situation."

With a few minor exceptions, we have therefore avoided being too specific in offering detailed implementation instructions. We have sought, rather, to paint a broad picture, including some relevant detailed information that will hopefully contribute to the understanding of those interested in forms management.

For those who share that understanding, forms management offers an opportunity to contribute significantly to the organization and cost effectiveness of any business. This will ultimately be reflected in the service that the customer receives, the cost of goods or services, the speed at which information reaches its destination, and the response it elicits, considerations which surely effect society as a whole?

****

PROFILE:

Jack Stubbs, Marketing Division, Moore Business Forms Inc., Glenview, Illinois.

Jack Stubbs has an association with DP-MA that behan in the early 1950s. While working as a data processing manager for Moore in Toronto, Canada, he was active in the formation and development of the Toronto Chapter of the DPMA.

He moved to Montreal in 1959 and played an active part in the Montreal Chapter for a number of years.

Now a member of the Marketing Division of Moore, Jack has continued his interest in DPMA affairs. In 1974, he addressed the Western Ontario Conference of the DPMA on the "Paper Crises".

DESCRIBING THE REAL WORLD FOR THE DATABASE

AUTHORS: M.M. Ridwan, B.Sc.(Hons)
Assistant Lecturer
Department of Computer Science
University of Canterbury
CHRISTCHURCH

and Z. Loy, B.E.(Hons), Ph.D., MBCS, MIEE
Lecturer
Department of Computer Science
University of Canterbury
CHRISTCHURCH

The importance of a framework for the description of the real world for the database has lately been recognized. A model for this purpose, called the data abstraction model, is presented in this paper.

The basic DATAM constructs are presented and the use of DATAM in the description of the enterprise view of data is discussed. The expressiveness of the data abstraction model is illustrated with examples and the transformations of a DATAM model to network and relational models are considered.

Key Words and Phrases: Enterprise view of data, data abstraction model, description of the real world, network model, relational model, database

CR Categories: 3.50, 3.70, 4.33, 4.34.

1 INTRODUCTION

Many database management systems are now available, e.g. TOTAL, IMS-DL/I, DMS 1100. These systems generally employ network, hierarchic or relational *data models* [5]. Data models such as these however, stress the structural representation of data objects rather than the representation of real world data concepts. Models that provide this latter data abstraction perspective will be termed *abstract models*.

Attention is being given to the abstraction of data (Ledgard and Taylor [7]). The aim of data abstraction is to model real world information situations from the point of view of the enterprise rather than from that of the computer system. Data abstraction is an important communication interface between the enterprise and D.P. personnel in the application of database management systems. An abstract model provides a framework for the description of the real world for the database.

The importance of a framework for the description of an enterprise view of data has lately been recognized (ANSI/X3/SPARC [1], Chen [3]). Such a framework provides the necessary interface for the definition of a process to produce a good database design. In this paper we present a method for the abstraction of real world data. DATAM, a generalized data abstraction model is introduced in Section 2. DATAM is based on definitions of formalized entity and relationship data concepts. Two examples in Section 3 illustrate the capability of DATAM in modelling semantic information of the real world. The correspondence between a DATAM model and the network and relational models is described in Section 4.

2 THE DATA ABSTRACTION MODEL (DATAM)

2.1 Abstraction Types

In the data abstraction model we identify the following (concept) abstraction types or database constructs.

1. *Entity* which models the existence of a real world object and its capability of being described.
2. *Attribute* which represents the concept of the description of an object where the descriptor object is not itself described.
3. *Relationship* which represents the concept of the mutual description or association between two entity or event types.
4. *Event* which models a describable object in which the object represents an association between two or more entity or event types.

Diagrammatic representations of these abstraction types and their associational connectives are given in Figure 1.

2.2 Entity

The most basic abstraction is that of the database entity (db-entity) representing a real world entity. A real world entity (ANSI/X3/SPARC [1]) is "a person, place, thing, concept or event, real or abstract, of interest to the enterprise". Within the database universe, an entity is perceived as a single, uniquely identifiable object. An entity set is a collection of entities of the same type.

Figure 2 illustrates a simple DATAM diagram for a "supply" model. The db-entity abstraction types shown for Project, Supplier and City represent the corresponding real world entity types of interest.

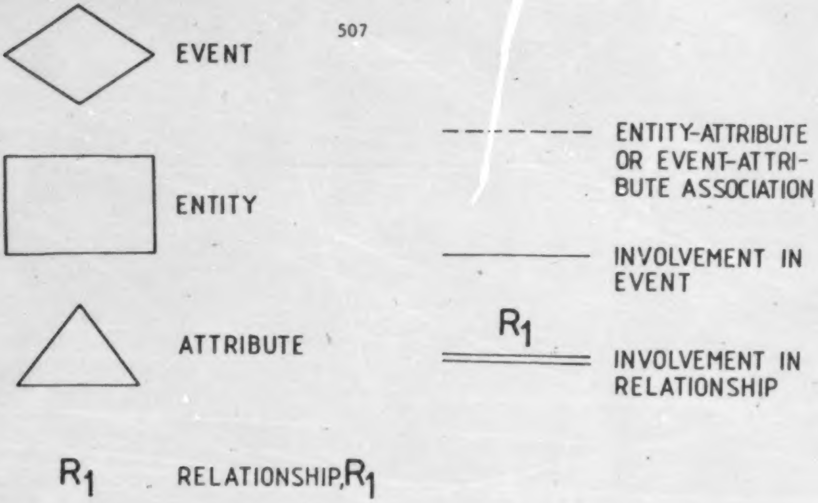

EVENT
ENTITY
ATTRIBUTE
R_1
RELATIONSHIP, R_1
ENTITY-ATTRIBUTE OR EVENT-ATTRIBUTE ASSOCIATION
INVOLVEMENT IN EVENT
R_1
INVOLVEMENT IN RELATIONSHIP

FIGURE 1

DATAM diagrammatic representations of concept abstraction types and their associations

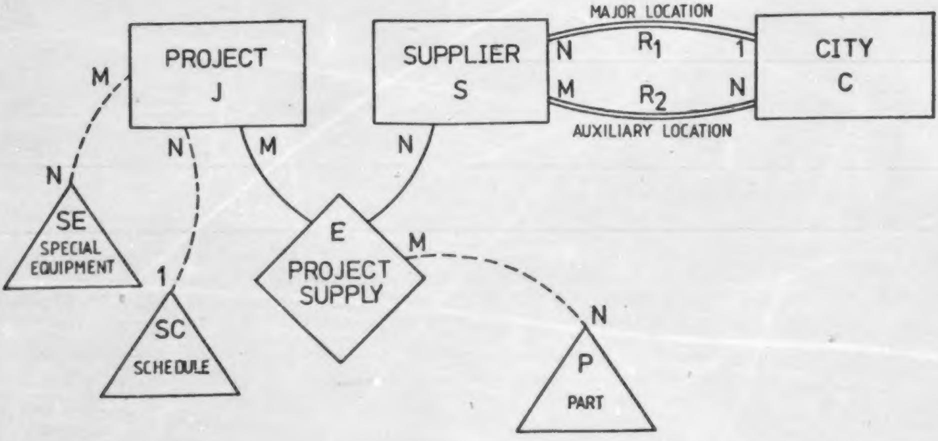

PROJECT
J
SUPPLIER
S
CITY
C
MAJOR LOCATION
R_1
AUXILIARY LOCATION
R_2
N
1
M
N
M
N
M
N
N
1
SE
SPECIAL EQUIPMENT
SC
SCHEDULE
E
PROJECT SUPPLY
M
N
P
PART

FIGURE 2

Project-Supplier-Part model

2.3 Attribute

Associated with an entity may be several *attributes*, representing the descriptions (properties, characteristics) of the real world object which the database entity is representing. An object is a property when it is being used to describe another object; it is an entity when it itself is being described. Thus in the database abstractions, a database entity is described by db-attributes or, equivalently, db-attributes describe a db-entity.

In Figure 2, Project is seen as an entity while Special-Equipment and Schedule are seen as attributes as they are not themselves described. The Project : Special-Equipment association, for example, represents the description "requires Special-Equipment" and indicates the association between instances of the Project entity to instances of the Special-Equipment attribute. This mapping or association between Project, J, and Special-Equipment, SE, is shown as *m:n*.

2.4 Relationship

In the modelling of the real world, situations often arise where two entities are related or associated together; e.g. a person may be described by his relationship with a car which is itself perceived as an entity. We call this a db-relationship.

Note that both attributes and relationships model the *concept* of describing. However, a relationship facilitates an association between two entities, while the association between an entity and its attribute is direct. Thus there may be many relationships between any two entities, that is, the entities may describe each other in many ways, while an attribute represents only a single conceptual description of a particular entity.

Figure 2 shows two relationships between Supplier, S, and City, C, illustrating a multiple relationship situation. S and C describe each other through the relationships. For example the $n: 1$ relationship, R_1, can be viewed as "Suppliers S have major locations in Cities C" or the inverse "Cities C are the locations for Suppliers S". The other $(m:n)$ relationship, R_2, between S and C indicates "auxiliary location".

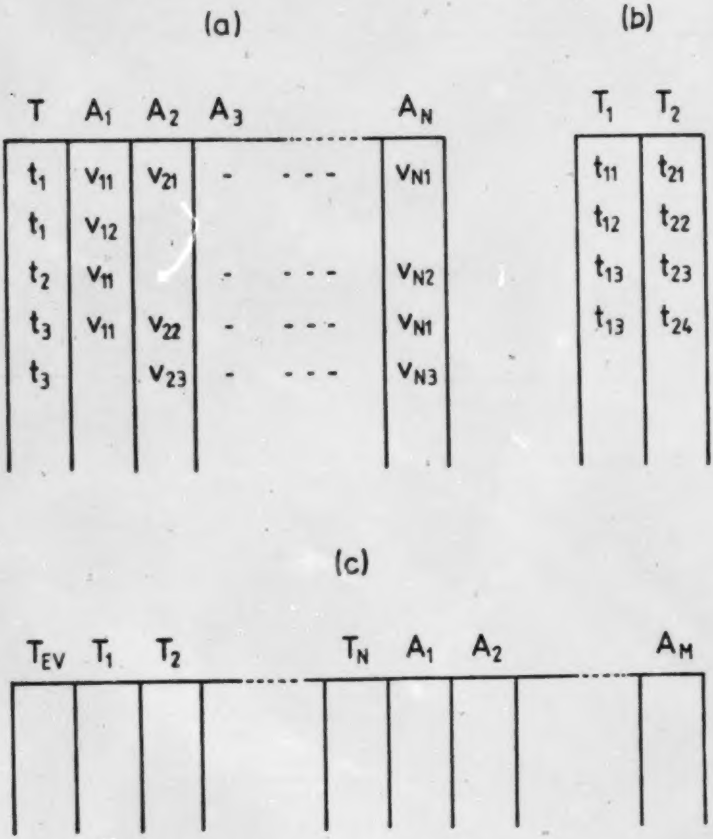
2.5 Event

The event represents the association between two or more (describable) objects where the association itself may be described. An example of an event is shown in Figure 2. The event, E, Project-Supply, involves the entities Project and Supplier and represents the perception of the $(m:n)$ relationship between Project J and Supplier S as a describable object. The event, E, is described by an attribute, Part, representing the Part supplied in the supply to J by S.

2.6 A Representation of DATAM Instances

While a DATAM diagram such as Figure 2 represents the description of the perceived objects (abstraction types) and their associations, it is sometimes necessary to have a perception of all the data and associational instances that are present. The representation of these instances is provided by the following tables.

2.6.1 EA-table type. Instances of the associations of an entity to its attributes are represented by an EA-table (Figure 3(a)). In this table, the first column represents a token instance set of an entity type. (A token value represents a unique object). The other columns represent the instance sets of all the attribute types of that entity. Each row of the table indicates a particular association of the entity instance to its attribute instances. For example, the first row in



(a)

T	A_1	A_2	A_3		A_N
t_1	v_{11}	v_{21}	-	- - -	v_{N1}
t_1	v_{12}				
t_2	v_{11}		-	- - -	v_{N2}
t_3	v_{11}	v_{22}	-	- - -	v_{N1}
t_3		v_{23}	-	- - -	v_{N3}

(b)

T_1	T_2
t_{11}	t_{21}
t_{12}	t_{22}
t_{13}	t_{23}
t_{13}	t_{24}

(c)

T_{EV}	T_1	T_2		T_N	A_1	A_2		A_M

FIGURE 3. DATAM instance representations

(a) entity-attribute, EA-table type
(b) relationship, R-table type
(c) event, Ev-table type

Figure 3(a) represents an association of a token t_1 of the entity to particular attribute instances, e.g. v_{11} of attribute type A_1. Row two represents an association of the same token t_1 with another instance, v_{12}, of the attribute A_1.

2.6.2 R-table type. Tables of this type represent relationships. Each table (e.g. Figure 3(b)) consists of two columns containing token instances of the two participating object-types (entities or events).

2.6.3 Ev-table type. These tables represent events. Each table has columns representing the event tokens, tokens of the member objects, and the attribute instances of the event. These represent the existence of the event and the descriptions of the event. An Ev-table is illustrated in Figure 3(c).

3 DESCRIBING THE REAL WORLD DATA CONCEPTS

3.1 The Modelling Process

In the description of the real world data concepts, it is necessary to distinguish between the universe of real world objects and the universe of database objects. The modelling of the real world in database terms centers around the modelling of the existence and description of objects of the real world and the association between these objects. To illustrate the modelling process, two examples are presented here. These examples show that, as opposed to the network, hierarchic or relational models, the DATAM model reflects more closely the enterprise's view of its data resource.

3.2 Project-Supplier-Part Model

The "supply" situation for an enterprise is to be described. The perceived data and their associations are:

1. Suppliers supply Parts to the various Projects being undertaken by the enterprise.
2. The various Projects may require Special Equipment.
3. There is a Schedule associated with each Project.
4. The Suppliers each have a major location (City) and may also have auxiliary locations in other Cities.

Figure 2 shows the representation of this situation. The Project-Supply event describes the supply of Parts to the Projects by the Suppliers. Each Project has attributes, Special-Equipment and Schedule. The major and auxiliary locations for the Suppliers are indicated by the two relationships between the Supplier and City entities.

3.3 Stock-in-Department Model

A second example of the description of an enterprise situation is presented in Figure 4. For the enterprise, the location of its Departments is indicated by the Dept:City relationship. The Population for each city is represented as an attribute of the entity, City. The Quantity of each Stock item held by a Dept is represented as an attribute of the event, Stock-in-Dept. The price of each Stock item is represented as an attribute of the entity, Stock.

4 TRANSFORMATION TO DATA MODELS

4.1 DATAM as a Semantic Reference

DATAM provides a framework for the description and representation of real world data concepts and a database system based on the DATAM concepts may be constructed.

However, a DATAM diagram may be transformed to other models. Thus the expressiveness of DATAM may be exploited by using DATAM in the description of the enterprise view

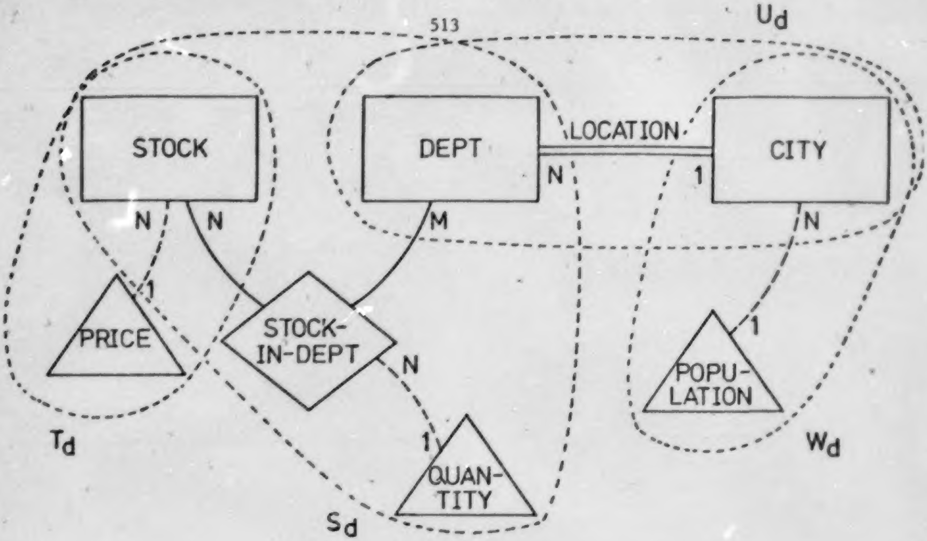

513
Ud
STOCK
DEPT
LOCATION
CITY
N
1
N
N
M
N
1
PRICE
STOCK-IN-DEPT
N
1
POPULATION
Td
1
QUANTITY
Wd
Sd

FIGURE 4
Stock-in-Dept model

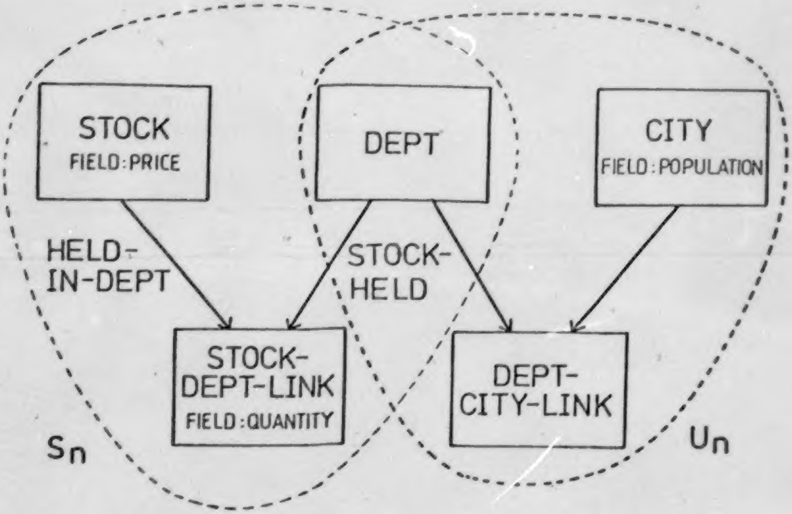

STOCK
FIELD: PRICE
DEPT
CITY
FIELD: POPULATION
HELD-IN-DEPT
STOCK-HELD
STOCK-DEPT-LINK
FIELD: QUANTITY
DEPT-CITY-LINK
Sn
Un

FIGURE 5
A Network Stock-in-Dept model

of data. This DATAM diagram is then transformed to the model of the available database management system for representation and storage purposes. In this case, however, the semantic aspects inherent in DATAM is not explicitly incorporated in the resulting model and becomes an external responsibility.

The transformations of the DATAM model in Figure 4 to the network and relational models are illustrated in this Section.

4.2 Network Model

The network model (Computing Surveys [5]) corresponding to the DATAM model of Figure 4 is shown in Figure 5. The DATAM entities, STOCK, DEPT and CITY become record types and their attributes become non-key fields within the record type. The event Stock-in-Dept is transformed into an owned record type, Stock-Dept-Link.

The relationship between Dept and City also becomes an owned record type, Dept-City-Link. The dotted lines in Figures 4 and 5 indicate corresponding substructures in the models.

4.3 Relational Model

The relational model (Codd [4]), involves the construction of normalised relations. The transformation of the DATAM diagram in Figure 4 to a relational model results in the following set of normalised relations:

S (STOCK#, DEPT#, QUANTITY)
T (STOCK#, PRICE)
U (DEPT#, CITY)
W (CITY, POPULATION)

These relations correspond to the substructures indicated in Figure 4.

5 CONCLUSION

The trend in database systems is to provide frameworks for the description of information situations from the enterprise view (Biller and Neuhold [2], Kerschberg et al [6]). Data models, such as the network and relational models, fall short of this objective. As a result, abstract models are being developed to provide better modelling facilities.

A data abstraction model has been presented for the modelling of data for database systems. The expressiveness of DATAM is illustrated by examples. The transformations of a DATAM diagram to the network and relational model are also presented.

6 ACKNOWLEDGMENT

One of us, M.M. Ridwan, acknowledges the support of a New Zealand University Grants Committee Postgraduate Scholarship.

7 REFERENCES

[1] ANSI/X3/SPARC, Study Group on Data Base Management Systems: Interim Report, ANSI, 75-02-08. In *FDT, ACM-SIGMOD 7,* 2 (1975).

[2] BILLER, H., and NEUHOLD, E.J. Concepts for the conceptual schema. In *Architecture and Models in Data Base Management Systems,* G.M. Nijssen, Ed., North-Holland Pub. Co., Amsterdam, 1977, pp.1-30.

[3] CHEN, P. P-S. The entity-relationship model - a basis for the enterprise view of data. Proc., AFIPS 1977 NCC, Vol. 46, AFIPS Press, Montvale, N.J., pp.77-84.

[4] CODD, E.F. A relational model of data for large shared data banks. *Comm. ACM 13, 6* (June 1970), 377-387.

[5] COMPUTING SURVEYS, March 1976. Special issue on database management systems. *Computing Surveys 8,* 1 (March 1976), 1-151.

[6] KERSCHBERG, L., KLUG, A., and TSICHRITZIS, D. A taxonomy of data models. In *Systems for Large Data Bases,* P.C. Lockemann and E.J. Neuhold, Eds., North-Holland Pub. Co., Amsterdam, 1976, pp.43-63.

[7] LEDGARD, H.F., and TAYLOR, R.W. Two views of data abstraction. *Comm. ACM 20,* 6 (June 1977), 382-384.

DIRECT OR INDEXED SEQUENTIAL? - A FILE DESIGNER'S GUIDE

AUTHOR: OWEN HANSON

The City University Business School,
London,
United Kingdom.

ABSTRACT:

Direct and Indexed-sequential files are compared to decide when it is appropriate to use each file organisation technique. Space requirements are examined, and methods of optimising the performance of each type of file are given. Data on when to update an indexed-sequential file directly, and when to batch and sort input, are also given.

KEYWORDS & PHRASES:

Sequential, indexed-sequential, blocking, hit rates, run times, block hit rates, block size, record formats, skip sequential processing.

C.R. CATEGORIES

3.70, 3.72, 3.73, 3.74, 3.79, 6.34

DIRECT OR INDEXED SEQUENTIAL? - A FILE DESIGNER'S GUIDE

Owen Hanson

The City University Business School, London, United Kingdom.

1. Introduction

Indexed-sequential file creation and processing software is readily available both from computer manufacturers and software houses. The user programmer will tailor the software to meet the needs of any given application, and may opt for one type of IS software rather than another, but that is the extent of the work he has to expend in implementing an indexed-sequential file. Directly organised files, by contrast, require individual design; as sets of keys differ one from another, generalized packages cannot handle many of the decisions that are required. The result of this difference is that indexed-sequential files are often used when direct files would be more appropriate. This paper assesses the two file organisation techniques comparatively, to provide a basis for correct, rather than convenient, choice between them.

This choice is clearcut in extreme cases. A genuinely mixed application, requiring considerable direct-access and significant sequential processing, calls naturally for an indexed sequential file. On the other hand, an enquiry-only situation in which a hgih volume of throughput is expected is best handled by a directly organised file. (However, if the enquiry volume is relatively low, it may be desirable to use an indexed sequential file, both for flexibility of access and for more convenient security arrangements).

The area of uncertainty includes both choice between the two file organisations and the mode of processing for an indexed sequential file. A very small number of updates is best handled by direct processing, whatever file organisation is used. At some point, however, it is better to batch and sort input rather than process it directly. Methods of determining the break even point are discussed below.

2. Space Requirements

a. Indexed-Sequential Files.

In comparison with a sequential or serial file, which consist only of data records, more space will always be required for an indexed file. The factors affecting this will be

(i) Record Format. In IBM-compatible systems, the record key has to be held separate from the body of the data, to allow search-on-key operations. This leads to a reduced storage capacity for data per track and per cylinder, which is reflected in increased space requirements. Table 1 shows the track storage capacities for records of varying sizes on an IBM 3330, using the separate key formats. The increased space requirements given indicate that, for medium sized blocks, a penalty of about 10% in space is paid for using the separate key format. Small records are more wasteful, while large blocks eliminate the problem unless record size is unfortunate (see the final entry in Table 1).

(ii) Indices. Most indexed-sequential software requires at least a track index (IBM) or equivalent for every cylinder; in addition a cylinder index (IBM) or seek area index (ICL) is always necessary. Higher-level indices are sometimes used, but generally take up little space in comparison with overall file requirements.

The cylinder index has to be stored on a separate cylinder, and it is often helpful to hold it on a different device, as this cuts down head movement time, and may also improve searching speed if a fixed-head disc or drum is used. However, the space required is generally a complete cylinder; only very large files require more.

Track indices take up between half and a full track of each cylinder. IBM Systems use two index records per track, each in separate key format, with a ten byte data field. The key length is determined by the user within the range 1-256 bytes, and a large key size will increase the proportion of the first track required by the index. On a 3330, for instance, a 128 byte key length would cause the whole first track to be taken up by the index, while a key length of one byte would allow data to be stored on almost 40% of the track.

(iii) Overflow areas. If additions are expected to an indexed file, overflow areas have to be allocated. The usual choice is: a cylinder overflow area consisting of one or more tracks per cylinder; an independent overflow area comprising a number of tracks (and often for convenience a full cylinder) separated from the main file area; a combination of cylinder and independent overflow areas. In assessing the storage space required for an indexed-sequential file, the options of no additions and of one track per cylinder plus a single cylinder independent overflow area will be analysed.

(iv) Packing density. Some manufacturers, including ICL, allow files to be only partly filled with records when they are created. This effect can only be achieved by the use of dummy records in cases where the software writers have not provided the appropriate coding. If such an option is used, it will be specified in the file design process, and will increase the space required to hold the data.

b. Direct Access Files

The factors that affect the space requirements of a directly organised file are as follows:

(i) Format. In this case records can be stored with keys either separate or embedded. If records are held singly, the separate key format allows a search-on-key if required. A block of records, on the other hand, is made up of records that randomise to the same address, but have nothing else in common. In these circumstances it is not advantageous to hold a single key separate to represent a block, so the embedded key format is used, saving space.

(ii) Packing density. When records are randomised to addresses, it is inevitable that some addresses will be allocated more than one record[1]. The top line of Table 2 shows the percentage of records that are synonyms, and thus are not stored in their home addresses, as a function of the packing density. As these records will require more than a single search to locate them, they slow up the processing of direct files. The designer will therefore wish to limit them, and will not exceed a packing density of 85-90% in most cases. This means that the file area allocated will be larger than that which could theoretically hold the file.

(iii) Bucket size. If records are randomised to buckets that can hold several records, far less synonyms - ie. records not held in their computed addresses - will occur than if single record storage positions are used. This is shown in Table 2. These buckets are, in effect, blocks of records. Storage in multi-record buckets thus improves the capacity of the file storage area, as well as reducing the number of misplaced records.

c. Comparison.

The space requirements of direct and indexed-sequential files are very similar. For comparison purposes, the space required for a 400,000 record file, of eighty bytes per record and with six byte keys, has been calculated for

(i) A sequential file.

(ii) A direct file, loaded in random order.

(iii) A direct file, access frequency loaded, and assuming that the most-used twenty percent of the records account for eighty percent of the transactions.

(iv) An indexed-sequential file in which overflow is expected, and for which one track per cylinder and one independent overflow have been provided.

(v) An indexed-sequential file for which no overflow is expected.

The figures are presented for single records, blocks of 3, 10, 25, 53 and a full track. The device is an IBM 3330. It is clear that, apart from the results for single records, there is little to choose between the space requirements of direct and indexed files. Even the disadvantage shown by direct files in this case can be avoided by storing single records, but randomising only to a track. A higher packing density can then by achieved, which reduces the space required.

Access frequency loading, which is fully described in [2], is the process of loading the most frequently referenced records first, and so ensuring that they are nearly all in their home address positions. The reduction in references to synonyms can be very marked, and allows densely packed direct files to be used without paying a heavy penalty. The packing densities shown have been chosen for convenience, and a designer might decide on different values in any given case.

The overall conclusion here is that disc storage space availability will not usually be a determining factor in choosing between the two types of file organisation. On the contrary, the designer generally decides on a file organisation, and then optimises the use of direct access storage for that organisation.

3. Direct Access Speeds

Both types of file will handle an individual enquiry quickly enough to satisfy the needs of the enquirer; even a directly-accessed sequential disc file will do this, as a response time of, say, five seconds is adequate. However, the throughput of enquiries possible in a given period will depend on the average access time for the file.

a) Indexed-Sequential Files.

(i) Index Position. Ideally the cylinder index should be held in the main computer storage, allowing very rapid reference, but limitations of main storage capacity usually prevent this. The next best solution is to hold the cylinder index on a separate device from that which holds the file. If a fixed-head disc or drum are available, this means that only a track search will be required, even if other active files are stored on the device. Failing this, a separate removable disc will often allow the access comb to be positioned over the cylinder index; by comparison, storage of the cylinder index and the main file on the same disc will lead to head movement for every access. Martin[3] has pointed out that, when the index and main file have to be stored on the same device, the cylinder index should be in the centre of the file area, as this minimises the number of cylinders to be traversed during head movements.

(ii) Software Effects. The timings considered in this paper are optimal; software implementations usually involve an additional time element, but as they can often by avoided by modification of the coding or adroit use of the options available, they will not be considered further here. Work which is proceeding to assess the losses involved will be reported in due course.

(iii) Optimum timings. Reference to any particular record involves the following elements:

STEP 1 Head movement to the cylinder index
STEP 2 Search of the cylinder index
STEP 3 Head movement to the data cylinder
STEP 4 Search of the track index
STEP 5 Search of the data track

If the cylinder index is held in main storage, the first two of these steps can be ignored, as they will be very rapid. This case, and also the other extreme of an average head movement plus one and a half revolutions of the disc to search the cylinder index, will be used in calculation. If the cylinder index exceeds three tracks - and hence the average search time exceeds 3/2 revolutions - a higher level index which acts as a track index to the cylinder index will be advantageous, and thus 3/2 revolutions is a reasonable value to use.

Step 3 will be taken to be an average head movement, while steps 4 and 5 will each be assumed to require half a disc revolution. This will hold true for all prime data track records, and for first overflow records from tracks; if a significant number of records are non-first overflows, the timings given here will be too low. However, they provide a useful basis for comparison.

For a 3330 the figures will be

STEP 1	30 milliseconds
STEP 2	25.1 milliseconds
STEP 3	30 milliseconds
STEP 4	8.4 milliseconds
STEP 5	8.4 milliseconds

Thus the time required to locate a record will be of the order of

102 milliseconds if the cylinder index is held on the same disc,

72 milliseconds if it is held on a separate, dedicated 3330, and

47 milliseconds if the cylinder index is in main storage.

These times will be reduced in some cases - for example, with a small file that has a central cylinder index on disc - but generally they will be optimistic, as software overheads and non-first overflow records slow up record retrieval.

b) Direct Access Files.

The location of a home record from a direct file involves only the calculation of its storage address from its key, which is carried out at processing speed, followed by a single head movement and a single track search. For a file taking up the whole of a 3330 disc pack, this would require 30 milliseconds
+ 8.4 milliseconds
= 39 milliseconds

(allowing 0.6 milliseconds for calculation of the address).

Synonyms require more time to locate, because the home record is reached first; when it has been retrieved and found not to be the required record, a further search is necessary. For this reason the file designer aims to minimise the number of synonyms occurring, and to reduce the impact of those that cannot be avoided. The factors he will consider include:

(i) Randomising Algorithm. Methods of allocating records to addresses were first discussed by Peterson[4], who defined open addressing. Buchholz[5] stated that division by a prime number was usually the most suitable technique for allocating addresses, and Lum et al[6] showed that for eight large files division produced less synonyms than would be expected from perfect randomisation of records to addresses. This was confirmed by Severance and Duhne[7], but Kaimann[8], who worked on a single existing file, did not obtain similar results, nor did Lin[9]. However, division will usually give numbers of synonyms similar to those predicted from table 2, and if the key sequences are favourable markedly lower synonym numbers may occur. If an algorithm is yielding significantly worse results than those shown in Table 2, it should be rejected.

(ii) Packing Density. The lower this is, the lower the percentage of synonyms will be. However, it is usually impractical to reduce synonyms by lowering the packing density below about 75%.

(iii) Bucket Size. As can be seen from Table 2, increasing bucket size smoothes out local overloads of records, and leads to lower synonym percentages. Within limits this is useful, but increasing bucket size means that larger blocks of records will have to be transferred into main storage, just to retrieve a single record. The block size will probably be limited, and transfer times may also become a significant factor. One way round this, albeit expensive in disc space, is to randomise only to a track, but store records on that track individually. This allows high packing densities because of the large 'bucket' capacity, but transfer time and record search time are kept low.

(iv) Loading order. When everything possible has been done to reduce the number of synonyms, it only remains to limit the impact of those that are unavoidable. If records are stored in random order, home records only should be stored on a first loading pass, and synonyms should be stored later. This 'two-pass load' minimises the number of synonyms. Whenever data about the frequency of use of records is available, the most used should be loaded first, employing the 'access-frequency load' technique described earlier. In this case synonyms and home records should both be loaded on the same pass, as it is not the number but the position of synonyms that is important. Files in which record 'popularity' varies widely, or with a short life, are not usually improved by access frequency loading.

(v) Overflow Handling. Methods of storing overflows have been discussed in [3,6,7], and figures have been given for the average numbers of accesses to synonyms. However, these authors did not explicitly consider the time of retrieval, and hence recommended storage of overflow records in a separate file area for cases where this reduces the average number of accesses to synonyms, but increases the average record location time. Both factors have been considered in [10]. It is shown that chained or tagged overflow storage techniques are better than open overflow for high packing densities and low bucket size.

c) Comparison.

After the precautions discussed above have been taken, a well designed file should not show an average additional time due to synonyms of more than 0.2 revolutions per record retrieved, and usually far less. This leaves the direct file access at less than 43 milliseconds, still some way better than the fastest possible indexed-sequential retrieval. If a direct file does not outperform an indexed file, then it should be re-designed. In general, direct files are very much faster than indexed files, not least because they have a much smaller software overhead.

4. Sequential Access Speeds

a) Indexed-Sequential Files.

In reading an indexed file sequentially, there will be a number of factors to be considered (see[1] for details)

1 Head movement to start
2 Track index entries are read for each track
3 Cylinder to cylinder transitions occur at the end of each cylinder.

Taking the case of 400,000 record file held on a 3330 that was mentioned earlier, and assuming ten record blocks, the 'overhead' costs of referencing an indexed sequential file would be:

30 milliseconds to start of file
177 x 10 milliseconds cylinder-to-cylinder transitions
1780 x 8.4 milliseconds track index references.

This gives a total of 16.75 seconds. There will be an additional 8.4 milliseconds rotational delay for each record referenced, and this is used to give figures for the average retrieval time of various numbers of records that have been batched for retrieval, rather than processed without batching and sorting.

The numbers of records involved in this sort of comparison will always be relatively low, as high hit-rate files that can be handled sequentially almost certainly will be. For this reason the actual sorting of a batch of records will be relatively rapid. However, the set-up time for a sort involves other operations, and sort times of 1 and 5 minutes have been added to the figures to give a realistic comparison. These figures are given in Table 4, and shown in Figure 1.

b) Direct Access Files.

Sequential retrieval from a direct file will be very much the same as direct retrieval. If an auxiliary file containing the actual addresses of every record in key sequence is available retrievals can be speeded up slightly, as no synonym effects occur. Otherwise, each key is transformed by the algorithm, and the record is retrieved normally

c) Comparison.

It is clear from Table 4 and Figure 1 that for very small groups of imput there is no advantage in batching and sorting. Ignoring sort times this becomes worthwhile for groups of 250-500 transactions, but the break-even point moves up to 1000-2000 transactions even with a one minute sort time. If sorting takes five minutes, this point is pushed up to between 3000 and 8000. The choice between direct and indexed-sequential files will depend on the possibility of storing the cylinder index in core. If this can be done, direct files have only a marginal advantage. If not, it is marked. The choice between the two types of processing of an indexed file may be equally dependent on index position.

Conclusions.

This paper has compared the attributes of direct and indexed-sequential files in order to indicate when each should be used. It has been shown that storage space is not an important factor, and that direct files should always provide a greater throughput of enquiries. If small batches of updates occur together, direct files remain superior, but the position changes if updates occur in larger numbers. The framework provided here is being tested in practice to assess the impact of software efficiency on the quoted results; first indications are that indexed files do not perform as well as is suggested here.

Security considerations may cause an indexed-sequential file to be used on occasions, as it is easier to check such a file rather than records which are held in a random order.

References

1. Designing Computer-Based Information Files, Owen Hanson, Pitmans, 1979.

2. Improving the Efficiency of Randomly Organised Files by Loading in Access frequency Order, by Owen Hanson, The City University Business School Working Paper, available from the Librarian, CUBS, 23 Goswell Road, London. EC1M 7BB

3. Computer Data-Base Organisation, James Martin, Prentice-Hall, 1975.

4. Addressing for Random Access Storage, W. W. Peterson, IBM Journal of R & D, April 1957.

5. File Organisation and Addressing, by Werner Buchholz, IBM Systems Journal, June 1963.

6.. Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files, Lum, Yuen & Dodd, CACM, April 1971.

7. A Practitioners' Guide to Addressing Algorithms, Severance & Duhne, CACM, June 1976.

8. Structured Information Files, Richard Kaimann, Melville Publishing Co., U.S.A., 1973.

9. Key Addressing of Random Access Memories by Radix Transformation, Andrew D. Lin, Proc. AFIPS 1963 SJCC, Vol. 24, Spartan Books, N.Y. pp 355-366.

10. Handling Overflows in Direct Files, Owen Hanson, Proc. ACS 8, Canberra, August 1978.

TABLE 1

Record Size, Bytes. Key Size is 6 Bytes	Record Format			Percentage of Extra Space Required	
	Embedded Key	Separate Key, Unblocked	Separate Key, Blocked	Unblocked	Blocked
80	61	48	47	27	30
160	44	37	36	19	22
240	35	30	30	17	17
320	28	25	25	12	12
400	24	22	22	9	9
480	21	19	19	11	11
560	18	17	17	6	6
640	16	15	15	7	7
720	15	14	14	7	7
800	14	13	13	8	8
880	12	12	12	Nil	Nil
2480	5	4	4	25	25

NOTES:

1. Blocked records take up more space than unblocked, because the highest key in the block is held both separately and in embedded form.

2. The final entry was chosen to give poor results for the separate key format. For large blocks the record size can be crucial, and format is usually not important. See the chapter on Record format in [1] for further details.

3. The figures tabulated under the 'Record Format' heading show the numbers of records of the stated size that can be held on one track of a 3330 disc.

TABLE 2

Records per Bucket	PACKING DENSITY, %																			
	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
1	2.46	4.84	7.14	9.37	11.52	13.61	15.63	17.58	19.47	21.31	23.08	24.80	26.47	28.08	29.65	31.17	32.64	34.06	35.45	36.79
2	0.16	0.60	1.29	2.19	3.27	4.49	5.83	7.27	8.78	10.36	11.99	13.65	15.34	17.03	18.73	20.43	22.11	23.79	25.44	27.07
3	0.01	0.09	0.29	0.63	1.13	1.80	2.63	3.61	4.73	5.99	7.35	8.82	10.37	11.99	13.66	15.37	17.11	18.87	20.64	22.40
4		0.02	0.07	0.20	0.44	0.80	1.30	1.96	2.78	3.76	4.88	6.15	7.54	9.05	10.65	12.33	14.07	15.86	17.69	19.54
5			0.02	0.07	0.18	0.37	0.68	1.12	1.72	2.48	3.40	4.49	5.73	7.12	8.63	10.26	11.98	13.78	15.64	17.55
6			0.01	0.02	0.08	0.18	0.37	0.67	1.10	1.69	2.45	3.38	4.48	5.75	7.18	8.75	10.44	12.24	14.12	16.06
7				0.01	0.03	0.09	0.21	0.41	0.72	1.18	1.80	2.60	3.58	4.74	6.09	7.60	9.25	11.04	12.93	14.90
8					0.02	0.05	0.12	0.25	0.49	0.84	1.33	2.03	2.90	3.97	5.23	6.68	8.30	10.07	11.96	13.96
9					0.01	0.02	0.07	0.16	0.33	0.61	1.02	1.61	2.38	3.36	4.55	5.94	7.52	9.27	11.16	13.18
10						0.01	0.04	0.10	0.23	0.44	0.79	1.29	1.98	2.88	3.99	4.80	6.86	8.59	10.48	12.51
11						0.01	0.02	0.07	0.16	0.33	0.61	1.04	1.66	2.48	3.53	4.36	6.30	8.01	9.90	11.94
12							0.01	0.04	0.11	0.24	0.48	0.85	1.40	2.15	3.14	3.98	5.82	7.51	9.39	11.44
14							0.01	0.02	0.06	0.18	0.30	0.57	1.01	1.65	2.52	3.64	5.03	6.70	8.54	10.60
16								0.01	0.03	0.10	0.19	0.39	0.74	1.28	2.05	3.09	4.41	6.00	7.85	9.92
18									0.02	0.06	0.12	0.28	0.55	1.01	1.70	2.65	3.90	5.45	7.28	9.36
20									0.01	0.04	0.08	0.20	0.42	0.81	1.42	2.30	3.48	4.99	6.80	8.88
25										0.01	0.03	0.09	0.22	0.48	0.93	1.65	2.70	4.10	5.86	7.95
30											0.01	0.04	0.12	0.29	0.64	1.23	2.16	3.47	5.18	7.26
35												0.02	0.06	0.18	0.45	0.94	1.76	2.98	4.64	6.73
40												0.01	0.04	0.12	0.32	0.73	1.46	2.60	4.22	6.30
50													0.01	0.05	0.17	0.46	1.04	2.05	3.57	5.63
60														0.02	0.09	0.30	0.76	1.65	3.10	5.14
75														0.01	0.04	0.16	0.50	1.25	2.59	4.60
100															0.01	0.07	0.27	0.83	2.02	3.99
150																0.01	0.09	0.43	1.37	3.26
200																	0.03	0.24	1.01	2.82

Note: The figures given in this table were first presented in Basic File Design, by Owen Hanson, published by IPC Business Press 1978. Figures tabulated show synonyms as a percentage of records stored, for a completely random allocation of records to addresses.

TABLE 3

FILE TYPE		Block or Bucket Size					
		One Record	5 Records	10 Records	25 (24*)	53 (52*)	Full Track
SEQUENTIAL	Tracks Required	6558	3810	2858	2667	2516	2470
	Cylinders Required	346	201	151	141	133	130
DIRECT, RANDOM LOADED	Key Format	Separate	Embedded	Embedded	Embedded	Embedded	Embedded
	Packing %	75	80	85	90	90	95
	Synonyms %	29.65	15.37	6.86	4.10	2.05	1.27
	Tracks Required	11112	4762	3363	2963	2796	2600
	Cylinders Required	585	251	177	156	148	137
DIRECT, ACCESS FREQUENCY LOADED	Key Format	Separate	Embedded	Embedded	Embedded	Embedded	Embedded
	Packing %	90	95	95	95	95	95
	% Synonyms Referred to	15.2	5.6	2.6	1.25	0.87	0.32
	Tracks Required	9260	4011	3009	2808	2649	2600
	Cylinders Required	488	212	159	148	140	137
INDEXED-SEQUENTIAL WITH OVERFLOW AREAS	Tracks Required	8334	4598	3077	2778*	2565*	2470
	Cylinders Required	479	265	178	161*	149*	144
INDEXED-SEQUENTIAL WITHOUT OVERFLOW AREAS	Tracks Required	8334	4598	3077	2778*	2565*	2470
	Cylinders Required	453	251	169	153*	141*	136

Note: The indexed-sequential file organisation technique requires a separate key record format, and 25 and 53 record blocks would be very wasteful of space. For this reason smaller block sizes have been used, which provide a more complete utilisation of disc space.

TABLE 4

Indexed-Sequential Files, Sequentially Accessed			
Number of Records	No. Sort	1 Minute Sort	5 Minute Sort
100	167.5	768.0	3168.0
250	67.0	307.0	1267.0
500	33.5	153.5	633.5
750	22.3	102.3	422.3
1000	16.8	76.8	316.7
1750	9.6	43.9	181.0
2500	6.7	30.7	126.7
5000	3.4	15.4	63.4
7500	2.2	10.2	40.9
10000	1.7	7.7	31.7

Direct Access Minimum	38.4
Direct Access Average	43.0
Indexed-Sequential Minimum	46.7
Indexed-Sequential, Cylinder Index on Separate Device	71.7
Indexed-Sequential, Cylinder Index on Same Device	101.7

Figures tabulated show the average time required to locate a record in milliseconds.

Note: the figures given for sequentially accessed indexed-sequential files here, and shown in Figure 1, do not include the 8.4 millisecond wait time for each record.

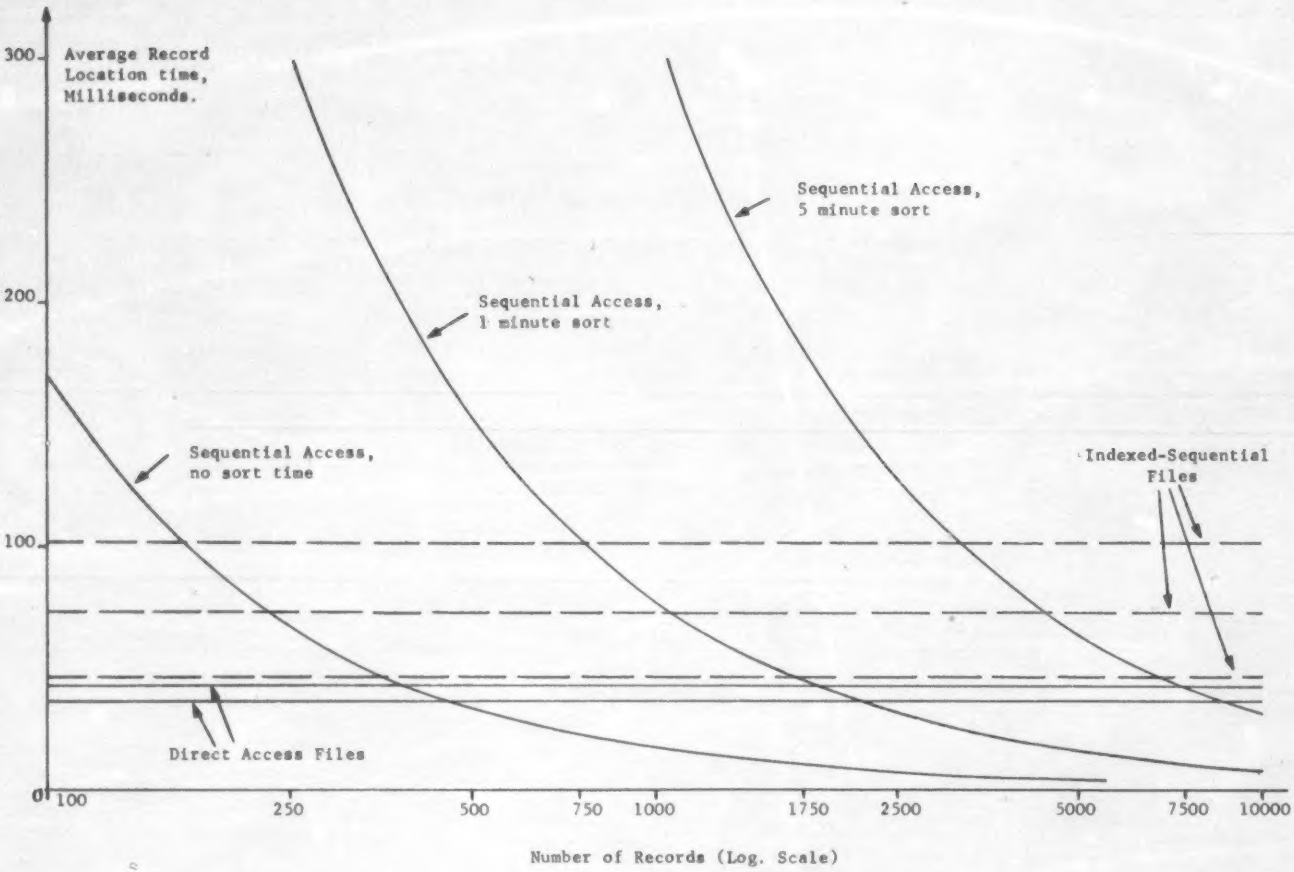

300
200
100
0
Average Record
Location time,
Milliseconds.
Sequential Access,
5 minute sort
Sequential Access,
1 minute sort
Sequential Access,
no sort time
Indexed-Sequential
Files
Direct Access Files
100
250
500
750
1000
1750
2500
5000
7500
10000
Number of Records (Log. Scale)

Figure 1